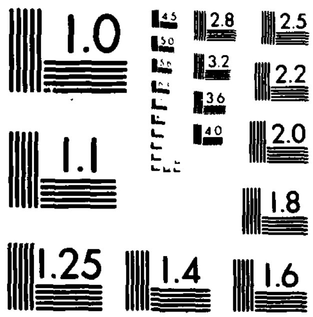
AD-A088 734 TEXAS UNIV AT EL PASO SCHELLENGER RESEARCH LABS F/G 4/1
THE DESIGN AND IMPLEMENTATION OF A COMPUTERIZED DATA REDUCTION --ETC(U)
JAN 80 K J HO, J D MITCHELL DAAD07-78-C-0010
UNCLASSIFIED SR3-80-UA-77 NL

1 of 2
AD
A088 734

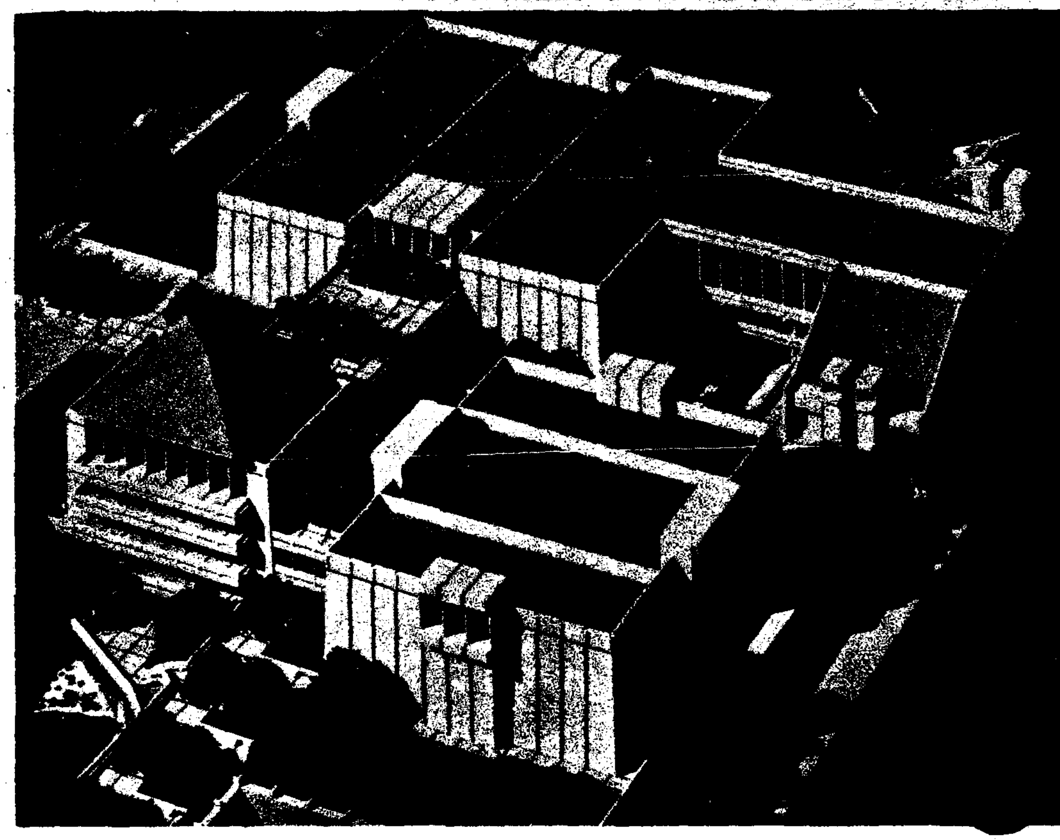

1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL

SCHELLENGER RESEARCH LABORATORIES

ELECTRICAL ENGINEERING DEPARTMENT

THE DESIGN AND IMPLEMENTATION OF A COMPUTERIZED DATA REDUCTION SYSTEM



ENGINEERING AND SCIENCE COMPLEX

The University of Texas at El Paso

THE DESIGN AND IMPLEMENTATION OF A COMPUTERIZED DATA REDUCTION SYSTEM

SPECIAL REPORT 01/31/80 SR3-80-UA-77

Prepared For:

United States Army Electronics Command

Atmospheric Sciences Laboratory

White Sands Missile Range

New Mexico

Submitted By:

Electrical Engineering Department

The University of Texas at El Paso

El Paso, Texas

Ka-Bun John Ho
Graduate Research Assistant

John D. Mitchell
Associate Professor

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (14) SR3-80-UA-77

2. GOVT ACCESSION NO.: AD-A088 734

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): ~~[illegible]~~ (6) The design and Implementation of a Computerized Data Reduction System

5. TYPE OF REPORT & PERIOD COVERED: (9) Special Report, ~~January 1980~~

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): (10) Ka-Bun John Ho ~~and~~ John D. Mitchell

8. CONTRACT OR GRANT NUMBER(s): (15) DAAD07-78-C-0010

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Schellenger Research Labs ~~Electrical Engr. Dept.~~ The University of Texas at El Paso, El Paso, Texas 79968

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (12) 134

11. CONTROLLING OFFICE NAME AND ADDRESS: (11) 31

12. REPORT DATE: Jan[redacted]80

13. NUMBER OF PAGES:

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release. Distribution unlimited.

18. SUPPLEMENTARY NOTES

Contract Monitor: Mr. Robert Rubio
Atmospheric Science Laboratory
White Sands Missile Range

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer, data reduction, middle atmosphere, electrical conductivity, rocket instrumentation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computerized system for reducing middle atmosphere electrical conductivity data obtained by blunt probes and Gerdien condensers is developed. In addition to improving the computational speed and accuracy, this particular system enhances the instrument's sensitivity range, i.e. the altitude region over which measurements can be obtained.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

1 8962

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The reduction system is designed strongly toward software portability and micro-processor adaptability. Electrical conductivity data from recently launched blunt probes were reduced using this system, and the results are discussed in this thesis. Future expansion, improvements and two proposed micro-processor system designs are also presented.

Accession For	
NTIS GRA&I	✓
DDC TAB	
Unannounced	
Justification per Letter on File	
By	
Distribution/	
Availability Codes	
Dist.	Avail and/or special
A	

ABSTRACT

A computerized system for reducing middle atmosphere electrical conductivity data obtained by blunt probes and Gerdien condensers is developed. In addition to improving the computational speed and accuracy, this particular system enhances the instrument's sensitivity range, i.e. the altitude region over which measurements can be obtained.

The reduction system is designed strongly toward software portability and micro-processor adaptability. Electrical conductivity data from recently launched blunt probes were reduced using this system, and the results are discussed in this thesis. Future expansion, improvements and two proposed micro-processor system designs are also presented.

TABLE OF CONTENTS

Table of Contents (continued)

Table of Contents (continued)

Table of Contents (continued)

LIST OF TABLES

LIST OF FIGURES

List of Figures (continued)

CHAPTER I

INTRODUCTION

1.1 BACKGROUND

The reduction of blunt probe and Gerdien condenser data from strip charts can be a relatively tedious and time consuming task. Also, maunal reduction techniques are limited in accuracy. With the advancements in computer technology and real-time programming techniques, this particular task is readily adaptable for a computer-assisted reduction method. In fact, such a method was initially attempted at The University of Texas at El Paso a few years ago with modest success [Shih (1977)]. Recently, this research effort was continued using state-of-the-art techniques which resulted in an operator-interacting , computerized reduction procedure for obtaining electrical conductivity from blunt probe and Gerdien condenser measurments. The presentation of this research is the subject of this thesis.

1.2 STATEMENT OF THE PROBLEM AND THESIS OUTLINE

The computerized data reduction system incorporating hardware design, software interfacing, programs and design strategy is presented in this thesis. The new system also provides data base management techniques for the user's convenience.

In Chapter 2, a brief description of the theory of operation for the blunt probe and the Gerdien condenser and the equations for calculating electrical conductivity are presented. A survey of former data reduction methods is considered in Chapter 3. Chapter 4 will discuss the hardware design for the data reduction system while Chapters 5, 6 and 7 describe the philosophy, design strategy, functionality and implementation of the system software. In Chapter 8, proposed micro-processor reduction systems are considered, one at the system level and the other at the component level. Finally, Chapter 9 is concerned with the error analysis of the data reduction system, and Chapter 10 presents results and considers future improvements to this new system.

CHAPTER II

BLUNT PROBE AND GERDIEN CONDENSER EXPERIMENTS

The blunt probe [Hale and Hoult (1965); Hale (1967); Hale, Hoult and Baker (1968)] and Gerdien condenser [Pedersen (1964); Rose and Widdel (1972); Conley (1974); Croskey, Hale and Leiden (1977); Mitchell, Sagar and Olsen (1977)] are instruments which measure electrical conductivity in the middle atmosphere. In addition, the Gerdien condenser measures ion mobility and charge number density. The payload is launched to a nominal altitude of 75 km using a meteorological rocket. Electrical parameters are measured after the payload separates from the rocket at apogee and is decending on a stabilized parachute. The probe's current-voltage response characteristics are telemetered to a ground receiving station and recorded on magnetic tape for later data reduction. A brief description of the instruments' operation is presented in the following sections.

2.1 CURRENT-VOLTAGE RESPONSE CHARACTERISTICS

The charged particle current collected by the probe's electrode is described by Ohm's law. For the blunt probe, the collector geometry is a disk. The equation for the charged particle collection current is:

$$\left| I \pm \right| = \frac{2r^2}{R} \sigma \pm \left| V \right| \tag{2.1}$$

The collection voltage waveform is a ramp which initially biases the collector at a negative potential with respect to the plasma, and then sweeps the collector to a positive potential. A complete sweep voltage waveform has a period of approxmately 8 seconds and nominal voltage limits of 5 V. In actual practice, the atmospheric electrical conductivity values are obtained by evaluating the derivatives of the probe's current-voltage response, thus avoiding the problem of having to estimate probe potential:

$$\sigma \pm = \frac{R}{2r^2} \left| \frac{dI\pm}{dV} \right| \tag{2.2}$$

The Gerdien condenser collector geometry consists of two concentric cylindrical electrodes. The collection voltage waveform, which biases the inner collector with respect to the outer return electrode, is a ramp voltage similar to the one described for the blunt probe. The resulting current response in the linear region of operation is described by the equation:

$$\left| I \pm \right| = \frac{2\pi \ell}{\ln\left(\frac{r_o}{r_i}\right)} \sigma_t \left| V \right| \tag{2.3}$$

The linear region of operation for a Gerdien condenser occurs at sufficiently low probe voltages such that no ion mobility group is completely swept out of the

air sample as it flows through the instrument. At larger probe voltages where one or more ion mobility groups are completely collected from the air sample, the current response displays additional structure that is used to calculate ion mobility and number density. The reduction procedure for determining these parameters is discussed by Pedersen (1964), Croskey (1976), Sagar (1977) and Domagalski (1979).

As was the case for the blunt probe, reduction of the Gerdien condenser data to determine electrical conductivity actually involves measuring of the derivative of the probe current with respect to voltage:

$$\sigma \pm = \frac{\ln\left(\frac{r_o}{r_i}\right)}{2\pi \ell} \left| \frac{dI\pm}{dV} \right| \tag{2.4}$$

2.2 DATA FORMAT

Prior to launch, the probe's current response with a known resistance value inserted between the collector and return electrode is received through the telemetry system and recorded on magnetic tape. This pre-flight calibration current response to the known collection voltage waveform is later scaled and used in reducing the in-flight conductivity data. The in-flight data are also telemetered back to ground, preferably to the same receiving station, and recorded on magnetic tape.

The probe's modulation scheme involves converting the electrometer's analog output signal to a pulse train which has a frequency proportional to the measured current. The relatively low frequency pulse train (nominally 0 - 200 Hz) modulates the grid of the 1680 MHz transmitter. This particular modulation scheme is compatible with the Meteorological Rocket Network's GMD-3 receiving system and TMQ-5 strip chart display system. In playing back the recorded data for reduction, the first step involves continuous measurment of the telemetered pulse frequency.

For this particular modulation scheme, the equations for conductivity as functions of pulse frequency are written for the blunt probe and Gerdien condenser as follows:

$$\sigma \pm = \frac{R}{2r^2} \frac{1}{R_{CAL}} \frac{(df/dt)_{DATA}}{(df/dt)_{CAL}} \quad \text{(BLUNT PROBE)} \tag{2.5}$$

$$\sigma \pm = \frac{\ln (r_0/r_i)}{2\pi \ell} \frac{1}{R_{CAL}} \frac{(df/dt)_{DATA}}{(df/dt)_{CAL}} \quad \text{(GEDIEN CONDENER)} \tag{2.6}$$

In these expressions, the pulse frequencies f_{CAL} and f_{DATA} correspond to the telemetered waveforms received during preflight calibration and while the instrument is in flight, respectively.

CHAPTER III

DATA REDUCTION METHODS

From the equations developed in Chapter 2, it is seen that reducing the blunt probe and Gerdien condenser data requires measuring both $(df/dt)_{DATA}$ and $(df/dt)_{CAL}$. Various methods have been used to obtain these values. Two former methods which will be discussed are the strip chart method and the computerized technique. In addition, the new computerized data reduction method will be introduced.

3.1 STRIP CHART METHOD

The strip chart method is probably the most used method for conductivity data reduction. The recorded waveforms are played back through a frequency-to-voltage transducer from which an analog signal is obtained and displayed on a strip chart. The waveform's slopes are then scaled and measured. From these measurments, the conductivities can readily be computed using, in addition, the physical parameters of the probe and the calibration waveform measurments. Examples of data waveforms displayed on strip charts are shown in Figure 3.1.

In spite of the simplicity of this method, there are inherent disadvantages associated with it, primarily, because the strip chart recorder is a mechanical device. The pen's time response degrades the display and the

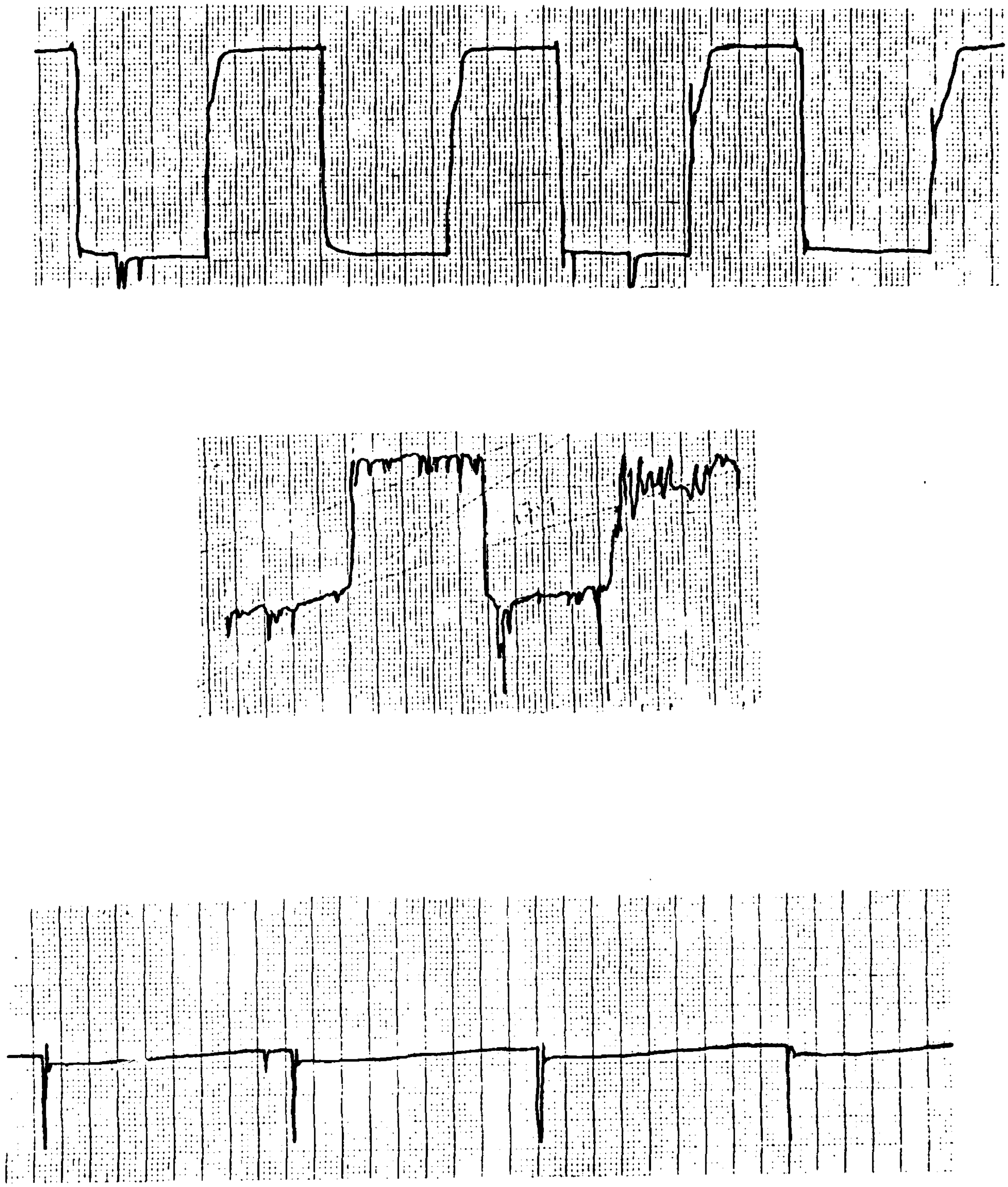


Figure 3.1 Data Waveforms Displayed on Strip Charts

scaling of waveforms obtained early in the flight is very difficult since the conductivity values, i.e. the waveform's slopes, are relatively large. At the other extreme, data obtained late in the flight have slopes so flat that scaling the waveforms introduces error. Finally, noise during the flight is usually exaggerated by the pen's inertia. The important advantage to this method is that a hard copy of the reduced waveforms does result.

3.2 FORMER COMPUTERIZED DATA REDUCTION METHOD

An earlier attempt was made to reduce conductivity data using a PDP 11/10 mini-computer with a Laboratory Peripheral System (LPS)[Laboratory Peripheral System User's Manual (1973)]. Computerized techniques enhance computational speed and accuracy, and they increase the altitude region over which the conductivity values can be extracted from the data waveforms. In the initial approach, however, there also were limitations.

First, the package was not designed with a uniform data structure. Thus, the data for each flight were considered individually and formatted differently. Secondly, the timing reference which was generated internally[1] would at times became distorted if signal dropout occurred for extended time intervals. Lastly, large por-

[1]Internal timing means the programmable real-time clock is used for both frequency measurment and time accmulation.

tions of the programs were written in assembler language, thus making it difficult to maintain or to modify. This also reduced protability between different machines.

3.3 IMPROVED COMPUTERIZED DATA REDUCTION METHOD

This research in computerized data reduction techniques not only improves the former system but in addition, it extends the capabilities with respect to portability of software, hardware design and data base management. The resultant product is a complete package that includes hardware design and software program modules to perform data acquisition, data reduction, data display and data base management.

3.3.1 BASIC PRINCIPLE

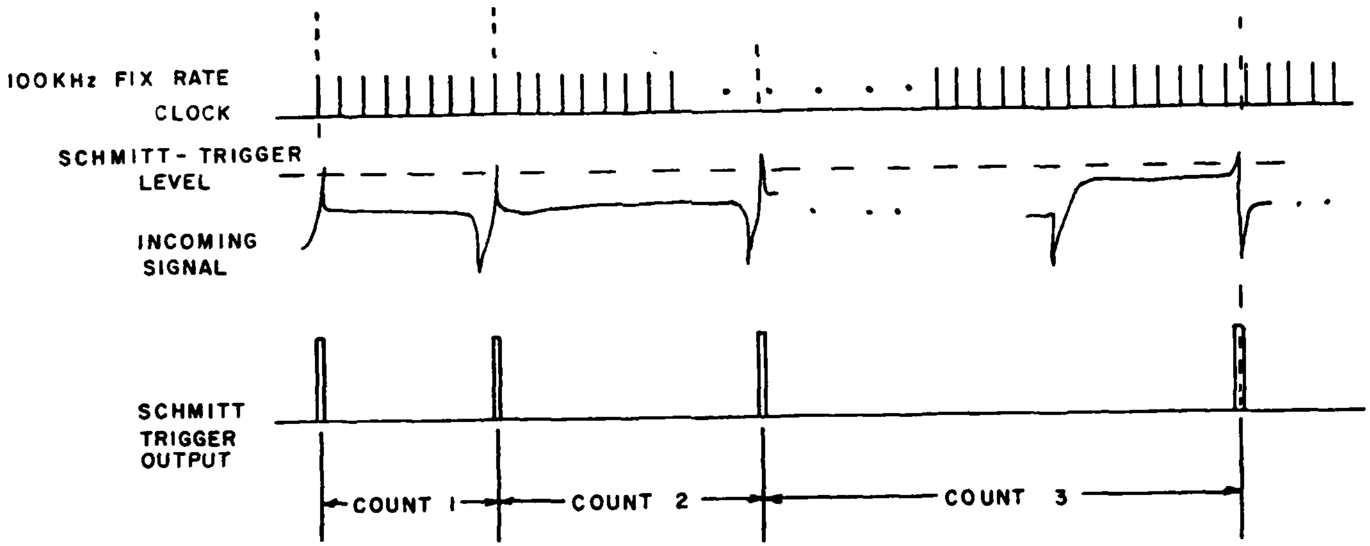
The basic concept of computerized data reduction is very simple. If the probe's transmitted signal , i.e. the waveforms, can be digitized, then they can be manipulated and processed by computers in a form suitable for reduction. Furthermore, they can be stored, retrieved and managed through mass storage media.

The method for digitizing waveforms is now explained. The incoming signal, which is in the form of variable frequency pulses, is compared against a relatively higher fixed frequency signal (clock). Each time a pulse is detected, the count from the fixed frequency

source is accumulated and referenced to the previously detected pulse. This count, which is a direct measurement of the time elapsed between two successive pulses, is then used to determine pulse frequency (Figure 3.2). This method is superior to using a combination of a frequency-to-voltage converter and an A/D converter to achieve the digitization.

After the waveforms have been digitized, they are stored on disk for further processing. Each waveform is then properly identified and accompanied by a header that contains relevant information including physical parameters of the instrument, the time and date of the launch and the timing of that waveform. These waveforms are written out to the disk as a file and stored for future processing.

The next step is scaling the waveforms. Instead of displaying the waveforms on a strip chart, the waveforms are shown on a storage scope. The operator identifies different slopes on the waveform and directs the computer to use numerical methods to calculate the best-fit straight line. The straight line that is calculated is generated on the screen to demonstrate that the fit is satisfactory. If the operator feels that the straight line is the proper one, the slope of the line is used for the calculation and the results are stored back in the header of each waveform. From this point on, the



$$\text{frequency} = \frac{100\,\text{KHz}}{\text{count}}$$

E.G. count = 500 frequency = 200Hz
count = 1000 frequency = 100 Hz

Figure 3.2 Waveform Digitization Method

flight consists of individual, self-contained waveforms, each complete with all information necessary.

Further processing includes plotting and printing of the data, data library creation and management, and transportation of the data to other computer systems.

3.3.2 MAJOR CHANGES

Three major changes have been included with the new computerized system for reducing electrical conductivity data. They are: inclusion of an external time code generator for referencing; implementation of a fixed data structure for computational flexibility; and programming in higher-level language for software portability. A discussion of these improvements is considered in the following sections.

3.3.2.1 EXTERNAL TIME CODE GENERATOR

The data reduction system uses an external time code generator for referencing the data waveforms. The programmable real-time clock of the LPS is thus dedicated to measuring pulse frequency.

When the payload launch is detected on the data tape playback, the time code generator is maunally initiated by the operator. For each data waveform, the operator issues a command via one of the A/D converter[2]

[2] A/D converter is analog-to-digital converter.

channels notifying the system to acquire timing information from the time code generator. This is done by sending hand-shaking signals via the Digital-input and Digital-output ports of the LPS through tri-state octal latches[3]. The particular time code generator in use sends out timing information in BCD[4] digits which are converted into binary form.

3.3.2.2 FIXED DATA STRUCTURE

A fixed data structure facilitates communication between program modules as well as transportation between different machines. Furthermore, it facilitates interfacing and future modification of the existing program modules.

Each waveform is accompanied by a 128 16-bit word header that contains all the physical parameters of the instrument as well as information about the flight. Conductivity values are included in the header after each waveform has been reduced.

3.3.2.3 HIGHER-LEVEL LANGUAGE PROGRAMMING

Most of the program modules are written in the higher-level language FORTRAN. This facilitates debugging and modification of the software. FORTRAN was also

[3]Tri-state devices can exist in an on, off and high-impedence state.
[4]BCD is Binary Coded Decimal.

chosen because of its popularity and I/O flexibility. The programs are coded to conform to STANDARD FORTRAN [PDP 11 FORTRAN Language Reference Manual (1974)]. Extensions to FORTRAN in the compiler are kept to a minimum.

Yet, some modules in the software package have to be written in assembler language, as in the cases of the interrupt service routine and real time processing, to quicken computation speed. However, the use of assembler language is kept to a minimum.

CHAPTER IV

HARDWARE SET-UP

The basic hardware for the data reduction system is discussed. The system's equipment is classified into four categories, each of which is described in this chapter. A complete hardware set-up schematic is shown in Figure 4.1.

4.1 COMPUTER SYSTEM

A processor with memory is required to execute programs, perform calculations and serve as temporary storage. Since the data reduction system works in an interactive mode, a console terminal is also necessary. A mini-computer or even a micro-processor will satisfy this requirement. In our particular system, a Digital Equipment Corporation (DEC) PDP™ 11/10 mini-computer with 16K words[5] of core memory, a RK05 disk cartridge and UNIBUS™ interfacing is used.

4.1.1 PDP 11/10 MINI-COMPUTER

The PDP 11/10 [PDP 11/04/05/10/35/40/45 Processor Handbook (1975)] is a relatively small computer in the DEC PDP 11 series of computers, but it shares the same architecture as the rest of its family (Figure 4.2). The processor is connected to the peripheral equipment (in-

[5] 1 Kword = 1,024 words.

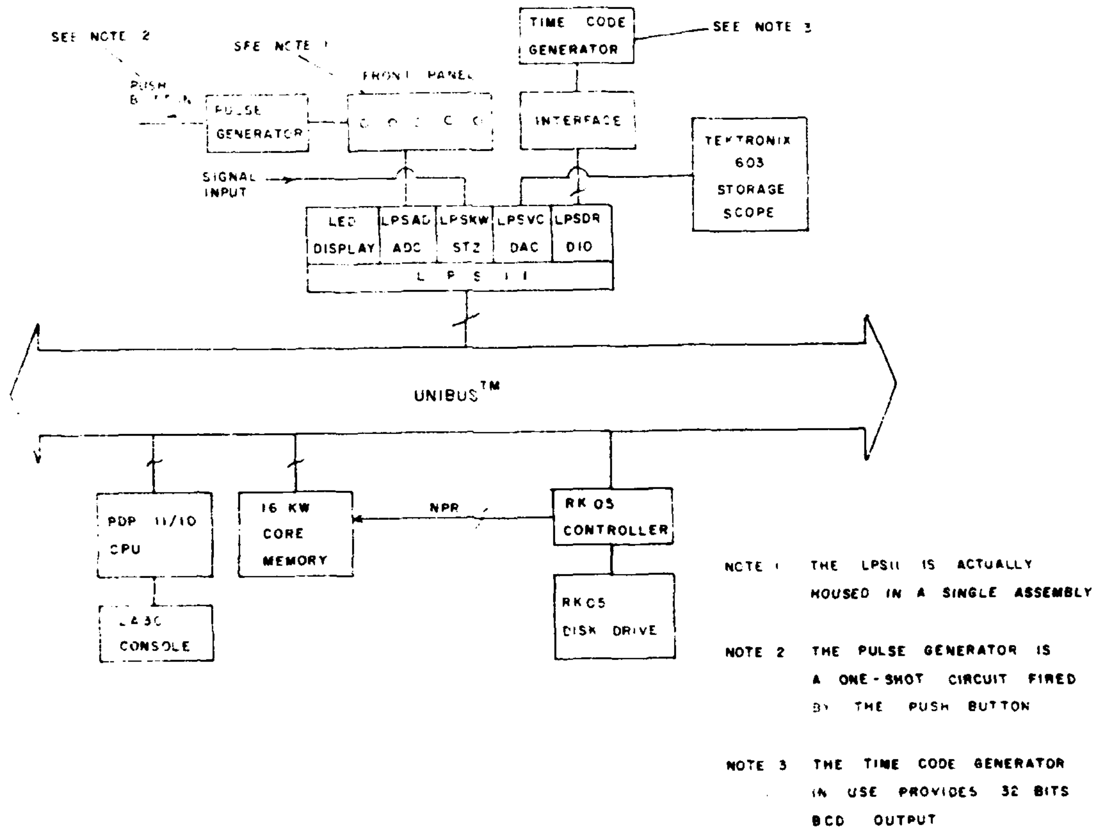


Figure 4.1 Hardware Configuration for the Data Reduction System

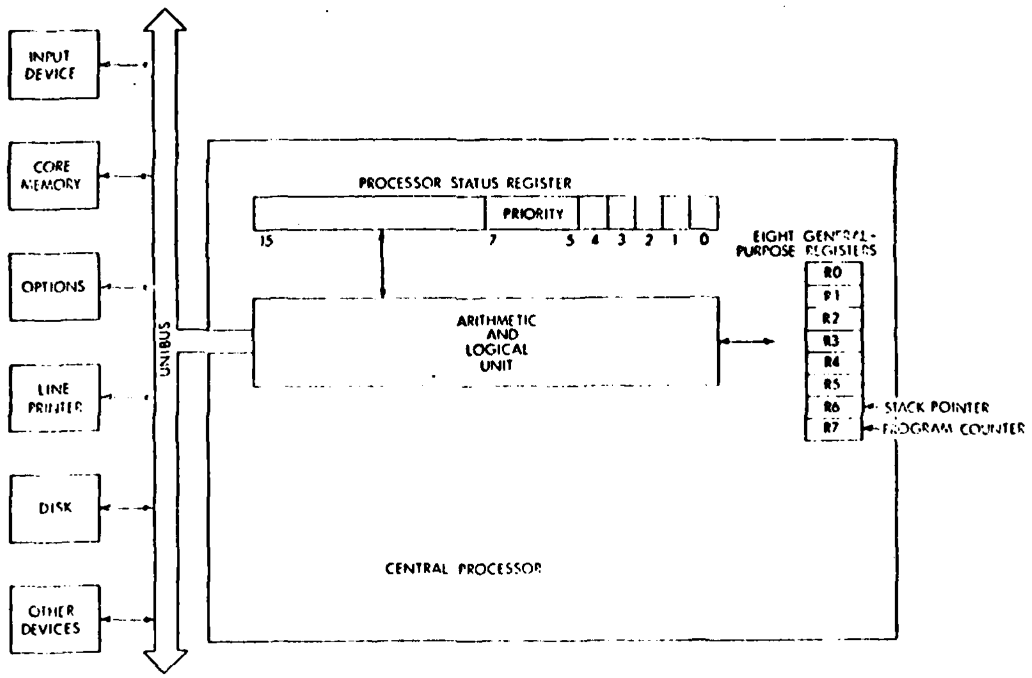


Figure 4.2 PDP 11 System Architecture

cluding memories) through a common set of wires called the UNIBUS. At any time, either one of the peripherals or the processor will have control over the UNIBUS (master) and will have access over any other members on the bus (slave). This kind of architecture makes interfacing of the processor to other equipment very convenient.

Another feature of the PDP 11 family of computers is that they provide priority vectored interrupt capability. Thus, external devices can interrupt the processor according to their pre-assigned priority, and control is then passed to a fixed location in memory called the Vector Address which is assigned to each type of device. As will be presented in later chapters, the interrupt mechanism is the basis of the data acquisition technique ; the ease in UNIBUS interfacing thus makes the hardware simple.

4.1.2 CORE MEMORY

Memory is required in any computer system to allow the storage of instructions and data. In our system, 16K words (each 16 bits) of core memory provide non-volatile storage for the system.

4.1.3 MASS STORAGE DEVICE

Input data eventually must be stored in some kind of mass storage device. Disks, drums, diskettes, or

tapes can serve this purpose. One consideration is the data transfer rate of the mass storage device which has to be compatible with the data acquisition rate. If not, some incoming data will be lost. Our system is equipped with a RK05 disk cartridge that can provide up to a 180K bytes/second data transfer rate and 2.5 million bytes of storage capacity [PDP 11 Peripherals Handbook (1975)].

4.2 WAVEFORM PROCESSING, MEASUREMENT AND DISPLAY HARDWARE

The clock, counter, Schmitt-trigger, A/D converter and display unit (Tektronix 603 storage scope) are the basic components for performing frequency counting and data display. The clock generates a fixed frequency reference for counting the frequencies of the incoming pulses. The counter accumulates the counts from the clock until a Schmitt-trigger is fired by the incoming data pulse and the count is then transfered to memory. The counter is, of course, cleared after the data transfer and it is then made ready for the next pulse.

The A/D converters serve two purposes. First, if the input data are fed to a frequency-to-voltage converter which in turn is connected to an A/D converter, a crude representation of the waveform can be displayed on a storage scope for waveform monitoring. Secondly, the A/D converters provide input ports for operator commands,

which can direct the program(s) to perform selective functions.

During the data acquisition phase, the display unit indicates the presence of an incoming signal. In the data reduction phase, the display unit displays the waveform images and allows the operator to identify and select proper slopes of the waveform for electrical conductivity calculations.

The clock, Schmitt-trigger, counter and A/D converters can be built using standard SSI and MSI[6] components. A storage scope or raster scan can serve as the display unit. In our system, however, a DEC Laboraory Peripheral System (LPS) and a storage scope are used to allow maximum design flexibility.

4.2.1 LABORATORY PERIPHERAL SYSTEM (LPS)

The LPS [Laboratory Peripheral System User's Manual (1973)] is a piece of OEM[7] equipment designed by Digital Equipment Corporation. It consists of a programmable real-time clock, eight A/D converter channels, two Schmitt-triggers, a display driver that can be directly connected to storage scopes, two digital I/O ports (one for the input, one for the output) and UNIBUS interface control logic for direct interfacing to the

[6] MSI and SSI are medium and small scale integrating circuit respectively.
[7] OEM is Original Equipment Manufacturer.

peripherals (Figure 4.3).

4.2.2 STORAGE SCOPE

The storage scope (Tektronix 603) [603 Monitor Instruction Manual (1972)] is connected directly to the display driver of the LPS and provides a 4096 x 4096 dot array plotting capability. Plotting and erasing are directly under software control.

4.3 PERIPHERAL EQUIPMENT

Additional peripheral equipment are also needed for this particular data reduction problem; namely, a time code generator, circuitry to interface the time code generator to the LPS and a pulse generator.

4.3.1 TIME CODE GENERATOR

A timing device is needed to reference the time elapsed from the launch of the rocket to the time of a particular in-flight data waveform. This in turn enables the operator to determine the instrument's position and velocity from the radar data. The timing device, after started by the operator at launch, is run asynchronously, independent of the rest of the system, to keep track of the time of flight. An Eldorado 1710 BCD time code generator is used for this purpose [Technical Manual, Model 1706 and 1710 Time Code Generator (1970)].

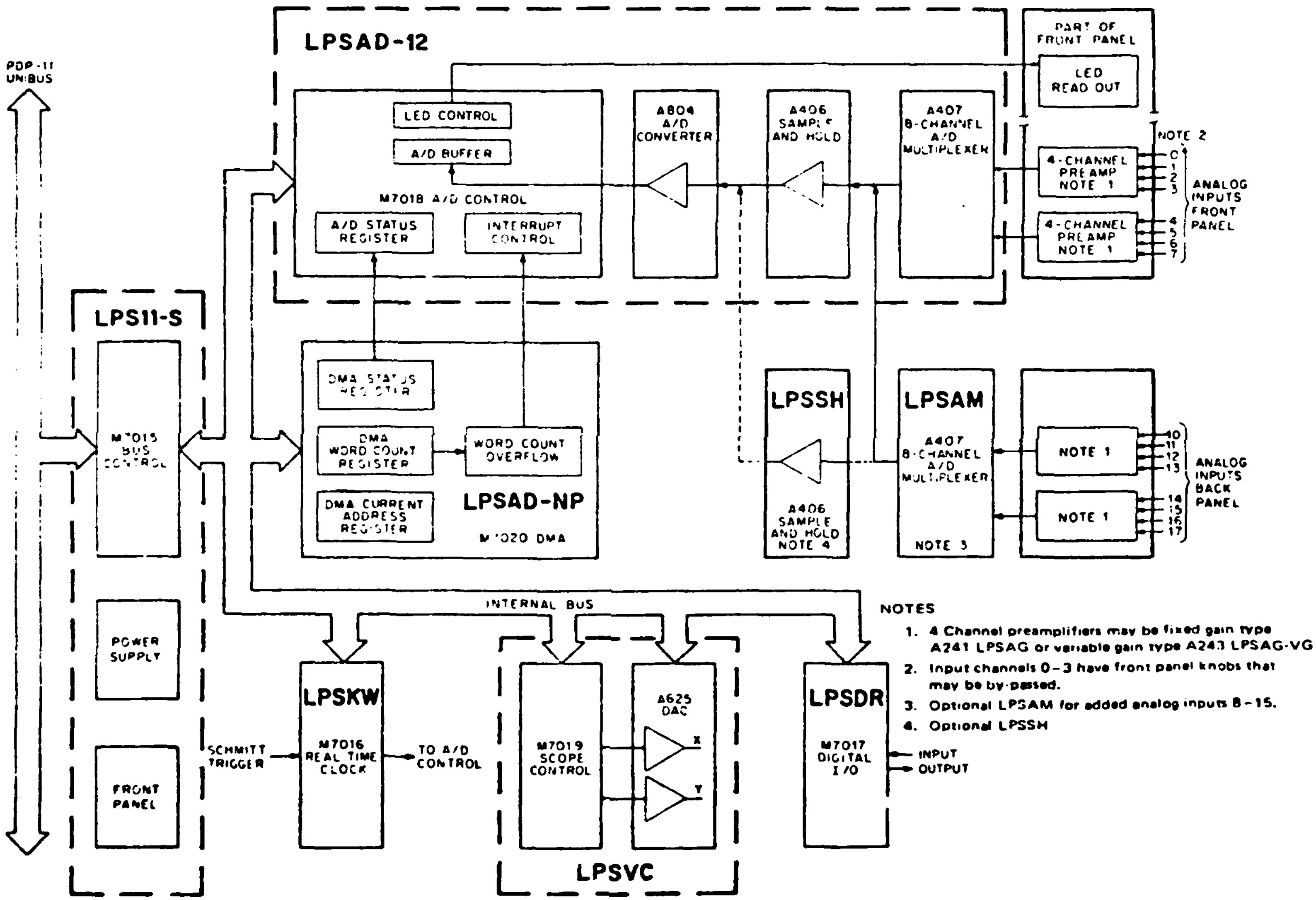


Figure 4.3 Laboratory Peripheral System Block Diagram

When a new waveform is recognized by the operator, a signal (a 5V rising pulse) is sent through one of the A/D converter channels to tell the data acquisition program that timing information is needed. At the same time, an endthenal[8] is padded to the data to allow a later program pass to segment the waveforms into individual units. Our timer provides 32 bits of BCD parallel output including seconds, minutes, hours, day, month and year. Only hours, minutes and seconds are needed for our application. Since it is not possible for the time code generator to input 32 bits in parallel form to the existing system, an interface has been designed to separate the signal into 4 bytes, 8 bits each. (This is explained in the following section.) By choosing one byte at a time, it is suitable for micro-processor interfacing since most second generation micro-processors are 8-bit machines.

The output of our time code generator is in BCD form and therefore must be decoded into binary form. It can be done very simply by software. If a binary output timer is used instead, decoding can be eliminated.

[8]An endthenal is a special character(s) to indicate the end of a record.

4.3.2 TIME CODE GENERATOR INTERFACE

In order to separate the 32-bit parallel output of the time code generator into 4 bytes of 8 bits each, four Intel 8212 octal latches are used to latch the timing information into the Digital input port of the LPS (Figure 4.4). The interface operates as follows:

> When the operator sees the beginning of a waveform from the display unit, a pulse is sent into a predescribed A/D converter channel (channel 0). This pulse tells the data acquisition program that a new waveform has come in and timing information is required. Under program control, a series of bits in the Digital output port of the LPS are set consecutively (Figure 4.5); they serve as hand-shaking signals for the time code generator interface to send in successive 8 bits of timing information in seconds, minutes, hours and day/year.

Four different bit positions of the Digital output port are tied to the chip enable (CE) pin on the four latches. Therefore each bit sent to the Digital output port enables a different latch to sample different timing information. Since the Intel 8212 octal latch is a tri-state device, all the latches that are not enabled will be in the high impedence state and look like open circuits to the rest of the system.

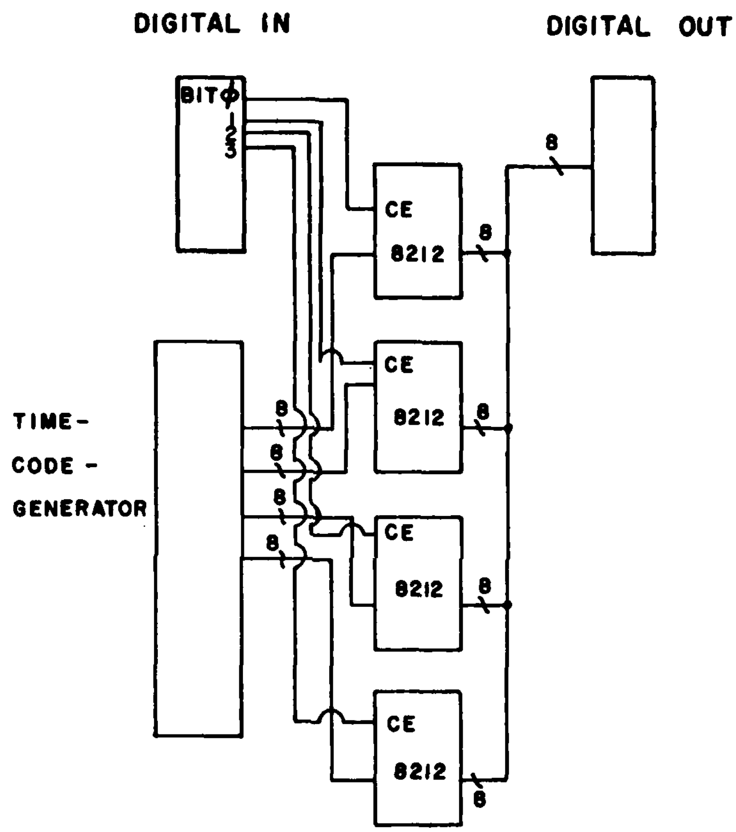


Figure 4.4 Time Code Generator Interface

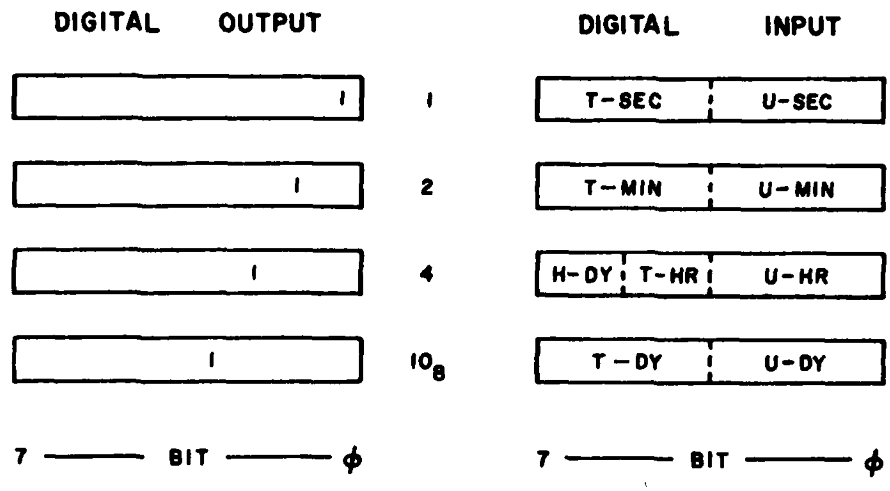


Figure 4.5 Hand-shaking Signals between the Time Code Generator Interface and the Laboratory Peripheral System

4.3.3 PULSE GENERATOR

The primary function of the pulse generator is to provide a fixed length pulse for the A/D converter as a command. Since the data acquisition program is essentially operating in a polling mode, the signal's pulse width must be controlled to avoid it from being carried around the loop(s) more than once and resulting in an error. A simple one-shot circuit is used and the pulse width is controlled by the RC time constant.

4.4 DATA PROCESSING EQUIPMENT

Processing of the reduced data, including analyzing, listing and displaying, is presently done using peripheral equipment on another computer system. Thus, such equipment is not essential for data reduction. Equipment used in this capacity are a PDP 11/45 minicomputer [PDP 11/04/05/10/35/40/45 Processor Handbook (1975)] operating under the UNIX Time-sharing System [UNIX Programmer's Manual (1975)] and supporting peripherals, which include a plotter, a storage scope, a line-printer, a 9-track industrial standard magnetic tape unit and a DECtape unit. The use of this second minicomputer system for data processing demonstrates the need for software portability.

CHAPTER V

DATA BASE DEFINITION

In this chapter and the following two chapters, the software design of the data reduction package is discussed. The description includes data base definitions, software design philosophies, implementation methods and individual program modules. From a present day's software engineering viewpoint, programs are combinations of data structures and algorithms [Wirth (1976)]. Without consistent data structures, program modules cannot communicate with each other efficently. A predescribed data base definition is essential for easy program modification and transportation. In this chapter, the data base definition of the different passes during the data reducton process will be examined, and their meanings and implications will be explained.

5.1 DATA STRUCTURE IN PASS 1

Pass 1 is the data acquisition phase. In this pass the incoming signal is digitized as previously described. The digitized data are stored under the file named "PASS1.TMP". It consists of records of variable length each representing a waveform in digitized form (Figure 5.1). When a falling pulse is detected at channel 0 of the LPS's A/D converter, a "0" (null character in ASCII) is put into the current buffer in memory. This

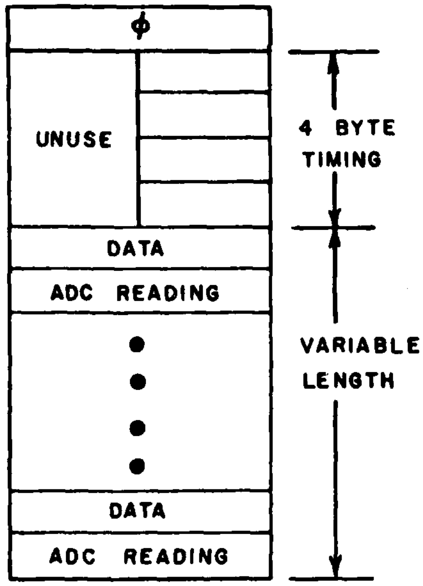


Figure 5.1

PASS 1 DATA STRUCTURE

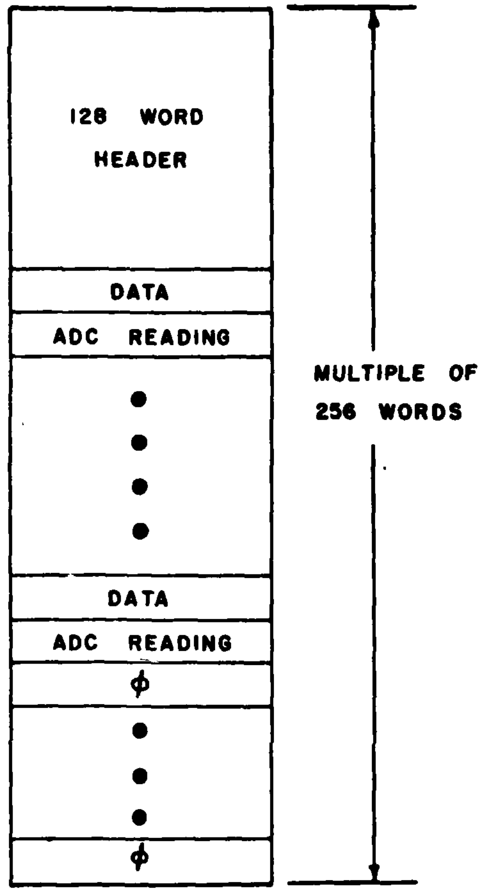


Figure 5.2

PASS 2 DATA STRUCTURE

null character serves as an endthenal to identify the beginning of a new waveform. It is then followed by 4 bytes of timing information obtained from the external time code generator. Following these 5 bytes are data points of the waveform in pairs. The first word is the count from the programmable real-time clock, and the next word is the reading from A/D converter channel 1. This word is not used at present but is reserved for future expansion of the system when multi-channel transmission or a special signal indicating the probe's zero-potential crossing may be stored.

Records are continuously written to the mass storage device until a falling pulse is detected at channel 2 of the A/D converter. This signal ends the data acquisition phase and the last buffer is filled with "107111_8" (ASCII representation of 'FI' for FINISH) to the next 256-word boundary. This last buffer is written out and the file is closed.

The data obtained in Pass 1 are primitive representations of the waveforms. The software of Pass 2 properly segments and refines the waveforms. The file "PASS1.TMP" can then be deleted.

5.2 DATA STRUCTURE IN PASS 2

The major function of Pass 2 is to refine and re-organize the raw data obtained from Pass 1. The data stream consists consecutively of a null, followed by 4 bytes of timing information and digitized data point pairs, which are now separated on a waveform-by-waveform basis. Each waveform is accompanied by a 128-word header and is rounded to the nearest 256-word block boundary (Figure 5.2). (A 256-word block is the basic unit used in RK05 disk cartridges. For different devices, this basic unit could be different.) This data structure is retained throughout the whole reduction process and serves as the data base in the package. The header, which contains all the vital launch parameters, will be discussed as follows.

5.2.1 HEADER

The header is 128 words long and is organized as shown in Figure 5.3. The information may be classified into two types: user supplied and program supplied.

During Pass 2 of the data processing, the operator is required to supply certain information requested by the program. This information includes: physical parameters of the probe; flight information such as launch site, launch date and launch time; and probe information such as the R_{CAL} and R_F values and the sweep

Word	Contents	Name	Word	Contents	Name
0	DATE	DATE(1)	32		
1			33	SPECIAL	
2	15 MAY 79	DATE(2)	34	IDENTIFICATION	
3			35	(cont.)	
4	SENSOR TYPE BP-G	STYPE	36		
5	SENSOR NUMBER	SNUMB	37		
6	LAUNCH SITE	LASITE	38	$T_{V\emptyset}$	TVØ
7			39	ZERO POTENTIAL TIME	
8	R_f	RF	40	ALTITUDE	ALT
9			41		
10	R_{cal}	RCAL	42	VERTICAL	VERVEL
11			43	VELOCITY	
12	r for BP	R 1	44	SATURATION CURRENT	ISAT
13	r_i for GC		45		
14	R for BP	R 2	46	NOT USED	
15	r_o for GC		47		
16	$\emptyset$ for BP	L	48	$\sigma+1$	SIG(8)
17	I for GC		49		
18	$\frac{df}{dt}$ cal	DFDTCL	50	$t_s\ \sigma+1$	TST(8)
19			51		
20	Δt SWEEP	DTSW	52	$t_F\ \sigma+1$	TFIN(8)
21			53		
22	ΔV SWEEP	DVSW	54	$V\sigma+1$	V(8)
23			55		
24	V SWEEP−	VSWN	56	$\sigma-1$	
25			57		
26	V SWEEP+	VSWP	58	$t_s\ \sigma-1$	
27			59		
28	SPECIAL		60	$t_F\ \sigma-1$	
29	IDENTIFICATIONS	IDEN(4)	61		
30			62	$V\sigma-1$	
31			63		

Figure 5.3

THE HEADER

Address	Content	Address	Content	Name
64	$\sigma + 2$	96	$\sigma + 4$	
65		97		
66	$t_s\sigma + 2$	98	$t_s\sigma + 4$	
67		99		
68	$t_F\sigma + 2$	100	$t_F\sigma + 4$	
69		101		
70	$V\sigma + 2$	102	$V\sigma + 4$	
71		103		
72	$\sigma - 2$	104	$\sigma - 4$	
73		105		
74	$t_s\sigma - 2$	106	$t_s\sigma - 4$	
75		107		
76	$t_F\sigma - 2$	108	$t_F\sigma - 4$	
77		109		
78	$V\sigma - 2$	110	$V\sigma - 4$	
79		111		
80	$\sigma + 3$	112	NOT USED	
81		113		
82	$t_s\sigma + 3$	114		
83		115		
84	$t_F\sigma + 3$	116		
85		117		
86	$V\sigma + 3$	118	ϕ	
87		119	ϕ	
88	$\sigma - 3$	120	TIME # 1	ITIME
89		121		
90	$t_s\sigma - 3$	122	TIME # 2	
91		123		
92	$t_F\sigma - 3$	124	N POINTS	NPTS
93		125		
94	$V\sigma - 3$	126	N BLOCKS	NBLK
95		127		

Figure 5.3 (continued)

voltage parameters. This information is put into the header and identifies the flight. The inclusion of this information makes each waveform a self-contained unit. If part of the flight data were ever lost, the remaining data would still be identifiable.

The second type of information in the header is supplied by the program, or more precisely, calculated by the program. Specific locations in the header are reserved to hold reduction results from Pass 3 of the data reduction package. The results include electrical conductivities for both positive and negative charged particles. The header also reserves room to hold more than one set of conductivity values, a possibility for Gerdien condenser data. Time information is obtained by decoding the BCD time code. This in turn enables one to determine the probe's position and velocity from the radar data.

5.2.2 OUTPUT FILE FORMAT IN PASS 2

After the data points have been segmented into individual waveforms and headers are inserted, the data points are rounded to the nearest 256-word block boundary by padding the remaining spaces with "0"s (zeros). Two other important parameters are then entered into the header; namely, the number of data point pairs and the length of the waveform in blocks. Then the waveform is

written out to a file called 'PASS2.TMP' and is ready for the next pass.

5.3 DATA STRUCTURE IN PASS 3

The software in the package is based upon the data structure developed in Pass 2. Pass 3 does not attempt to change any of the data points but merely displays the waveforms, scales the designated time segments, calculates the conductivity values and then inserts them into the proper locations in the header. Other utility programs then manipulate all the information as specified.

In addition to inserting values into the header of each waveform, Pass 3 also prints the information into an ASCII file and on the system console.

CHAPTER VI

SOFTWARE DESIGN PHILOSOPHIES

Three major concerns are carried throughout the design and implementation of the data reduction package; namely, modularity, portability and small system adaptability. These considerations not only influence the final form of the software but also the implementation methods as well.

6.1 MODULARITY

Modularity not only means "structure programming" [Dahl, Dijkstra and Hoare (1972)] as in terms of software engineering, but it also means a separation of tasks, each performed by different phases of the package. Although the programs are written partly in FORTRAN and partly in assembler language, discipline is enforced in coding to produce clean and readable code. Furthermore, subroutines and functions are used freely to allow program modules to be expressed in a structual manner in spite of the use of completely unstructured languages such as FORTRAN and assembler language.

In addition to clean coding, the data reduction package is composed of individual "PASSES", each of which performs a specific task that contributes to part of the reduction process. Designing the package in this manner makes each phase a small, manageable task. It allows fas-

ter coding and more importantly, easier debugging.

These passes communicate with each other through the use of files (data sets). The files are sets of data points which reside in mass storage devices, such as disk cartridges. There are advantages and also disadvantages associated with data file communication. The disadvantage is that it is much slower to communicate between files in a mass storage device than it is to put everything in memory. On the other hand, there are several advantages. First of all, work space is greatly expanded by providing virtual memory[9] capability to increase the work space. Secondly, tasks need not be carried out in a continuous manner, thus permitting a more flexible schedule. For example, if all the data reduction phases are not finished in one day, the remaining phases may be continued at some later time without the data being lost in the memory. Finally, data in files can readily be transported to other installations or computer systems for further processing or displaying. Data files in mass storage media such as disk cartridges or 9-track industrial standard magnetic tape make an attractive and cost effective method for transportation.

[9] Virtual memory means that disk space can be used for program or data space as if they are part of the memory available.

6.2 PORTABILITY

With the tremendous increase in software development cost, software portability has become a major concern in software engineering [Kernighan and Planger (1976)]. The hardware is becoming more sophisticated and inexpensive, and thus the use of higher-level language to implement software becomes feasible. The wide existence of FORTRAN compilers on computer systems makes FORTRAN, though far from the best, the almost universal programming language in engineering. Most of the data reduction package is written in ANSI FORTRAN, or more commonly known as STANDARD FORTRAN. The programs make minimal usage of the extension provided by our resident compiler. In cases where special system routines are used, a user-written subroutine could readily be substituted.

The use of higher-level language in programming allows more imagination in program design, more understandable coding and easier debugging and modification. In fact, during the design stage of the package, the thought that "We won't use assembler language unless necessary." was kept strongly in mind at all times.

Furthermore, the programs written are highly parameterized [Kernighan and Plauger (1976)], i.e. symbolic names are used instead of actual values. For example, if "BUFFERSIZE" is equated to 4096, all the programs

will use the symbol "BUFFERSIZE" instead of the number 4096. Later, if the buffer size needs to be changed to some other value, only the definition of "BUFFERSIZE" needs to be changed. This approach permits much easier modification and errors are less likely to occur.

In our installation, both programs and data have been transferred between the PDP 11/45, operating under the UNIX Time-sharing System, and the PDP 11/10 operating under RT-11. The implication is two-fold. First, the PDP 11/45 supports more peripherals and utility programs, like plotting routines, and thus can provide more attractive outputs. Secondly, it demonstrates portability between different computer systems and different operating systems. The results turns out to be satisfactory. In fact, a rough estimation shows that about seventy-five percent of the codes are portable. The other twenty-five percent belong mostly to the data acquisition phase which is machine dependent.

6.3 SMALL SYSTEM CONSIDERATION

The data reduction package is designed with small system adaptability in mind. This consideration is reflected in both the hardware design and the software design. For example, two future micro-processor systems are presented later in Chapter 8.

From the hardware aspect, all the required components are standard, easy-to-find, off-the-shelf SSI or MSI integrated circuits. These components can very easily be assembled into a unit that serves the same function as the LPS. The LPS, of course, offers much more flexibility in the design during the development and testing stages but in the production stage, a hard-wired unit could function just as well.

The data acqisition scheme is interrupt driven. The interrupt feature is supported both by mini-computers and micro-processors. Therefore, our data reduction system can be incorporated into any system by just rewritting Pass 1 of the data reduction package.

From the software aspect, two major concerns have been put into the design of the programs: optimzed program space and data space. In another words, the sizes of the programs are minimized rather than the speed. Small systems, either mini-computers or micro-processors, usually have only a limited amount of memory space as well as addressability. Accordingly, the programs also have to be small such that they do not take up too much room. This is first done by splitting tasks into separate passes and secondly, by carefully optimizing coding, even at the expense of using extra loops and subroutine calls. The data reduction process, however, is relatively slow compared to computer speed, and thus

not much is actually lost.

The other concern is minimizing data space. This is done by restricting array size and buffer size to levels compatible to smaller systems at the expense of extra I/O activities. In our particular system, asynchronous I/O is supported by the operating system and therefore problems of this nature do not arise.

CHAPTER VII

INDIVIDUAL PROGRAM MODULE DESCRIPTION

In this chapter, all the program modules in the data reduction package are examined. Their respective functions and implementation methods are discussed. In such cases that special techniques or tricks are used, they are pointed out for reference. The program modules can be classified into two categories; namely, the different passes and the utility programs. The different passes perform data acquisition, data segmentation and reduction while the utility programs do things such as print and plot results, extract data and put the data into different formats.

7.1 PASS 1

Pass 1 is the data acquisition pass. The incoming data are detected and digitized by the method described in Chapter 3. Its implementation combines polling and interrupt techniques. The polling scheme functions essentially as a big loop with the different A/D converter channels sequentially sampled to identify operator commands. The digitization of data is, however, done by firing of the Schmitt-trigger to interrupt the processor whenever a data pulse is detected. The programmable real-time clock LPSKW has the capability of saving the content of the counter in a register and of

allowing the interrupt service routine to store this count in memory. Therefore, data acquisition is interrupt driven.

The digitized data are written out to a mass storage device. Since I/O activity takes time, it is important to keep it at a minimum to reduce the loss of incoming data. Two methods can be used to accomplish this. The first method is to increase the buffer size for incoming data so that more data can be stored before each I/O activity takes place. However, this method was rejected because the package is aimed at micro-processor adaptation and for such systems, memory space is often limited. The second approach involves using a multi-buffer method. Whenever one buffer is full, the incoming data are directed to another buffer immediately while the first one is being written out.

Under the RT-11 operating system, asynchronous I/O is supported and can therefore further enhance the performance of a multi-buffer scheme. In our implementation, two buffers of 4K words each are designated for this purpose. A counter and a pointer are maintained by the program to keep track of the number of points in the current buffer and the location of where the next point is to be placed. When the current buffer is full, the counter is reset and the pointer designates the next buffer immediately. Simultaneously, the first buffer is

written out asynchronously while the normal data acquisition operation is resumed.

The LPS provides a digital input port and a digital output port which allow timing information from the time code generator to be latched in under program control instead of putting the signals on the UNIBUS directly. If this feature were not available, care would be needed in designing the time code generator interface to avoid damage to other peripherals on the UNIBUS.

It is necessary that Pass 1 be written in assembler language and that it be specifically constructed for a designated computer system since the interrupt mechanism and hardware configuration on various machines differ.

7.2 PASS 2

The purpose of Pass 2 is to separate the data obtained from Pass 1 into individual waveforms and to insert a header with each of them. The I/O for waveform segmentation makes use of a 4K-word input buffer and a 4K-word output buffer. The buffer size can be changed easily to fit different hardware configurations.

Because the output of the time code generator is in BCD, a routine to convert from BCD to binary numbers is required for decoding the timing information. This

particular routine can decode up to 1 hour 59 minutes and 59 seconds of timing code. The binary timing is also displayed on the LED output of the LPS to show the presence of waveforms and the relative timing. This pass is also written in assembler language. It could have been written in FORTRAN using binary read/write.

7.3 PASS 3

Pass 3 of the data reduction package is the most compilicated part of the whole system. It combines a display technique, a file management scheme and a numerical curve fitting method to allow the user: to examine the waveform; to expand a selected portion of it; to identify break points; to fit the best slope between them; to calculate results; and to store the results back into the header.

In order to assure adequate memory space, careful coding is used throughout the design of this pass. Due to the large number of routines in Pass 3, they are briefly classified according to their functions and discussed in the following subsections.

7.3.1 INPUT/OUTPUT ROUTINES

Input/Output activities are needed whenever a waveform is brought into display on the screen and then written out after reduction has been made with appropiate

information inserted in the header. In addition, an ASCII file is also created with the altitude and conductivities for quick reference.

Since waveforms are consecutively stored in multiples of 256-word variable length records, the sequential I/O approach is the most appropiate. The first block of each waveform is read in and the waveform length is found from the header. Then the remaining portion is read in and the complete waveform is either displayed or skipped, as determined by the operator. Although sequential I/O is slower than random access I/O, it is considerably simpler. Moreover, the sequential approach is appropiate when magnetic tape is used as the mass storage device.

The RT-11 operating system provides a system subroutine called 'ASSIGN' [PDP 11 FORTRAN/RT-11 Extansion Manual (1975)] which allows the user to assign logical unit numbers to devices and files. The programs heavily use this subroutine to communicate to physical devices such as disks and magnetic tapes. In other operating systems where this routine is not available, Job Control Language or user written subroutines can be substituted for it.

7.3.2 EXPANSION AND DISPLAY ROUTINES

After each waveform has been read into memory, it is displayed on the storage scope for viewing. The expansion routine and the display routine provide an accurate representation of the waveform on the scope and at the same time, they provide a convenient way for the user to identify end points for slope calculation.

The expansion and display routines work hand in hand to allow the operator to choose and display all or part of the waveform on the screen. The method and terminology used is confined to the CORE[10] standard proposed by the SIGGRAPH of ACM [Computer Graphics (1977)].

The Tektronix 603 storage scope provides 4096 x 4096 addressable points in both the X and the Y directions. Each point can be individually intensified to make up lines and curves. When each waveform is first read in, it is completely displayed from 0 to 200 Hz, which is the probe's nominal operating range. Then the operator decides if waveform expansion is needed for a more detailed viewing of selected part(s) (Figure 7.1). If expansion is needed, boundaries in both the X and the Y directions are entered by the operator through the console. An example is: "xmin = 100, xmax = 300, ymin = 60,

[10]CORE is the proposed standard in Computer Graphics from the Special Interset Group in Computer Graphics of the Association of Computation Machinary.

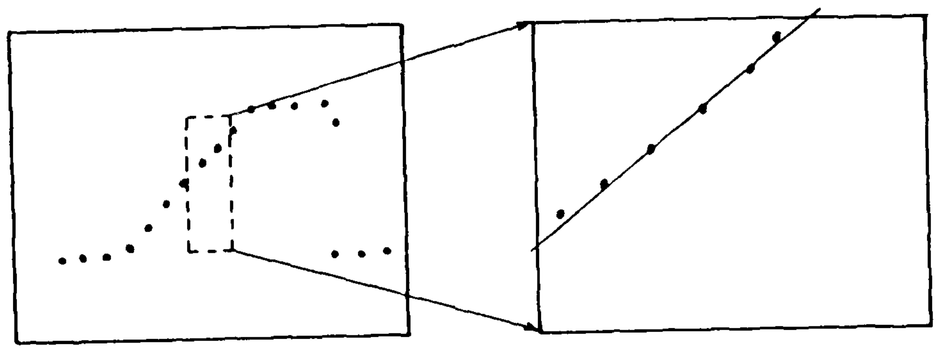


Figure 7.1 Expansion Routine Illustration

ymax = 120" which would be to expand the horizontal segment of the waveform from the 100^{th} point to the 300^{th} point, inclusive, and from 60 Hz to 120 Hz in the vertical direction. In CORE system's terms: the "world" contains the whole waveform; the "window" is set to [xmin,xmax] and [ymin,ymax]; the "screen" is the 4096 X 4096 screen surface with coordinates from 0 to 4095; the "viewport" is chosen such that it occupies the complete screen surface for viewing.

Other efforts have also been made to ease the reduction process. For example, both the horizontal and vertical axes are marked in grids and numbered. The horizontal axis is marked at every 50 points starting from the $xmin^{th}$ point. The vertical axis is marked at every 20 Hz starting from the $ymin^{th}$ Hz. Marking is done by drawing dashed lines across the screen at the appropiate locations. These spacings can be changed easily to suit individual taste. At the bottom of each vertical line and at the left of each horizontal line is a numeric string that indicates the value of the line. They are either frequencies in Hertz (vertical) or number of points (horizontal). Character string generation is done by a standard FORTRAN "encode" statement that converts from binary number representations into ASCII characters. The ASCII characters are then generated on the screen by 5x7 matrix points as shown in Figure 7.2. The size of

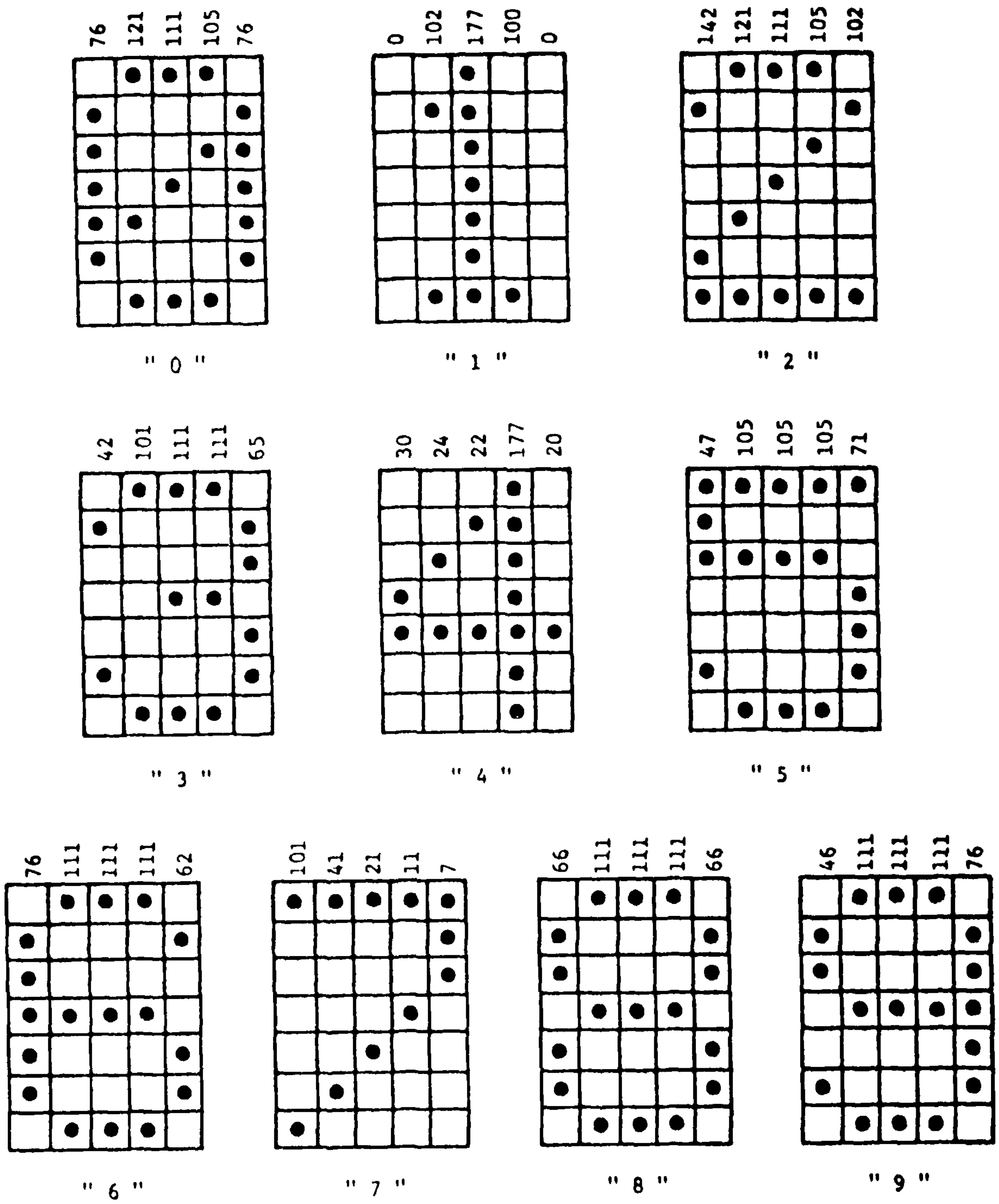


Figure 7.2 5 X 7 Matrix Representation for Digits

the characters can also easily be changed.

7.3.3 STRAIGHT LINE FIT AND LINE GENERATION

After each pair of end points on the waveform has been chosen by the operator, the best fit straight line to the data points between the designated end points is determined. The conventional least square method is used to determine the best straight line fit. In order to obtain more accuracy with the fit, an iteration of the least square method is made. First, all the data points between the two end points are assigned equal weights of 1.0 and a first fit is constructed with an equation in the form

$$f = Pt + Q \tag{7.1}$$

for which: f is frequency in Hz; t is time in seconds; P is the slope in Hz/sec and Q is the intercept in Hz. All data points are then weighted with respect to their absolute distances from the first fitted curve (Figure 7.3). A new weighting factor between 0.0 and 1.0 is assigned to each data point inversly proportional to this distance. Thus points farther away from the line have less influence in calculating the next straight line fit. A second iteration of this procedure again assigns new weighting factors to further refine the straight line fit. This iterative process results in a calculation of the waveform's slope over the designated time segment.

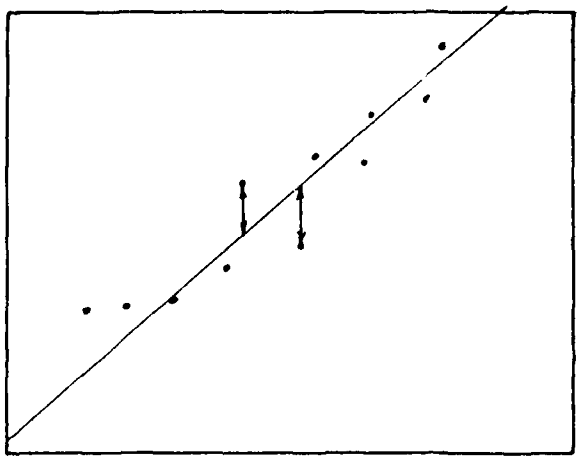


Figure 7.3 Weighting Factor Assignment

After the final values of P and Q are determined, the resulting straight line fitted to the data points is generated on the screen for the operator's examination. The line generation uses a Digital Differential Algorithm [Newman and Sproull (1979)] (DDA) for generating a series of points to making up the line. The line generation algorithm also determines the equation:

$$f = Pt + Q \qquad \text{or} \qquad t = (f - Q)/P \tag{7.2}$$

according to the ratio in the X and the Y directions of the window to obtain the most solid line possible.

If the operator determines that the straight line fit is not satisfactory, the process is repeated with other end points on the waveform designated. In cases where the waveform is so noisy that a good fit to the data is difficult to obtain, a very small time segment where the waveform is relatively clean can be selected to obtain a fit. The result is then examined and in most cases, the fit will be satisfactory.

7.3.4 ALTITUDE ROUTINE

Accompanying each data tape for each flight are usually radar data that provide time and altitude information. A routine has been written to interpolate the altitude of each waveform from the radar data. It is designed as a real function in FORTRAN that accepts time

as the input parameter and returns the exponentially interpolated altitude from the radar data. The radar data are taken from an ASCII file, thus allowing easy correction of the data.

7.3.5 OTHER BOOK-KEEPING ROUTINES

Other routines used in Pass 3 provide the following functions:

- inserting calculated results into the header;
- printing out results on the console;
- printing out results into an ASCII file for quick reference;
- reseting all parameters to their initial values.

7.4 UTILITIES

A number of utility programs have either been written or adapted to provide additional services to the user of the reduction package. These routines include:

- plotting data on the Tektronix 603 storage scope for quick viewing after reduction has been made;
- extracting conductivity and altitude values from the header of an already reduced flight into Basic-Plus virtual array (disk file) format for plotting on the PDP 11/45;
- printing out information from the headers, including launch date, launch site, physical parameters of the probe, time, altitude and conductivities;
- transfering files (including programs and data) between RT-11 and UNIX with the program "rtpip";

- creating data libraries and printing out programs with the program "PIP";

- using other system programs from RT-11 or UNIX.

CHAPTER VIII

MICRO-PROCESSOR BASED SYSTEM

The data reduction system is designed with micro-processor adaptability in mind. In this chapter, two possible micro-processor implementations are discussed. One is at the system level using Original Equipment Manufacturer (OEM) products while the other is designed at the discrete component level.

8.1 SYSTEM APPROACH

This proposed micro-computer system is based on the PDP 11/03 micro-computer, which is built around a LSI-11™ micro-computer [Microcomputer Handbook (1976)]. The LSI-11 is the smallest of the PDP 11 family, but it shares almost the same architecture and instruction set as the rest of the PDP 11 processors (Figure 8.1). The LSI-11 interfaces to the other peripherals through the Q-BUS™, which is very similar to the UNIBUS. The interrupt structure on the LSI-11 is the same as on the PDP 11, and therefore the same data acquisition method can be used. (However, on the LSI-11 only one level of priority interrupt is present instead of seven as on the PDP 11's. Our system only requires one level.) The block diagram of this system is shown in Figure 8.2 and the respective functions will now be discussed.

1	KD11-HA CPU	
2	MSV11 DD 32KW MEMORY	
3	RL01 CONTROLLER	
4	RL01 CONTROLLER	
5	DLV11-J SERIAL (4)	
6	BDV11 BOOTSTRAP	
7	(OPEN)	
8	(OPEN)	
9	(OPEN)	

Using the new 11/03-LK 9-slot backplane version of the PDP 11/03, the PDP 11T03-L System features the new RLV11 Disk Subsystem with two RL01 5.2 million byte cartridge disk drives (total 10.4 million bytes)

CONFIGURATION DATA:

Hardware: The system includes the PDP 11/03-LK microcomputer with 32K words (64K bytes) RAM, KEV11 Extended Arithmetic Option, BDV11 Bootstrap, two RL01 Disk Drives and Controller, H9612 Cabinet, and your choice of LA36 DECwriter II, VT100 CRT Terminal, or LS120 DECwriter III console device with EIA serial interface

Software: RT-11 Real-Time Operating System and a complete set of diagnostic programs for this configuration

Figure 8.1 PDP 11/03 System Configuration

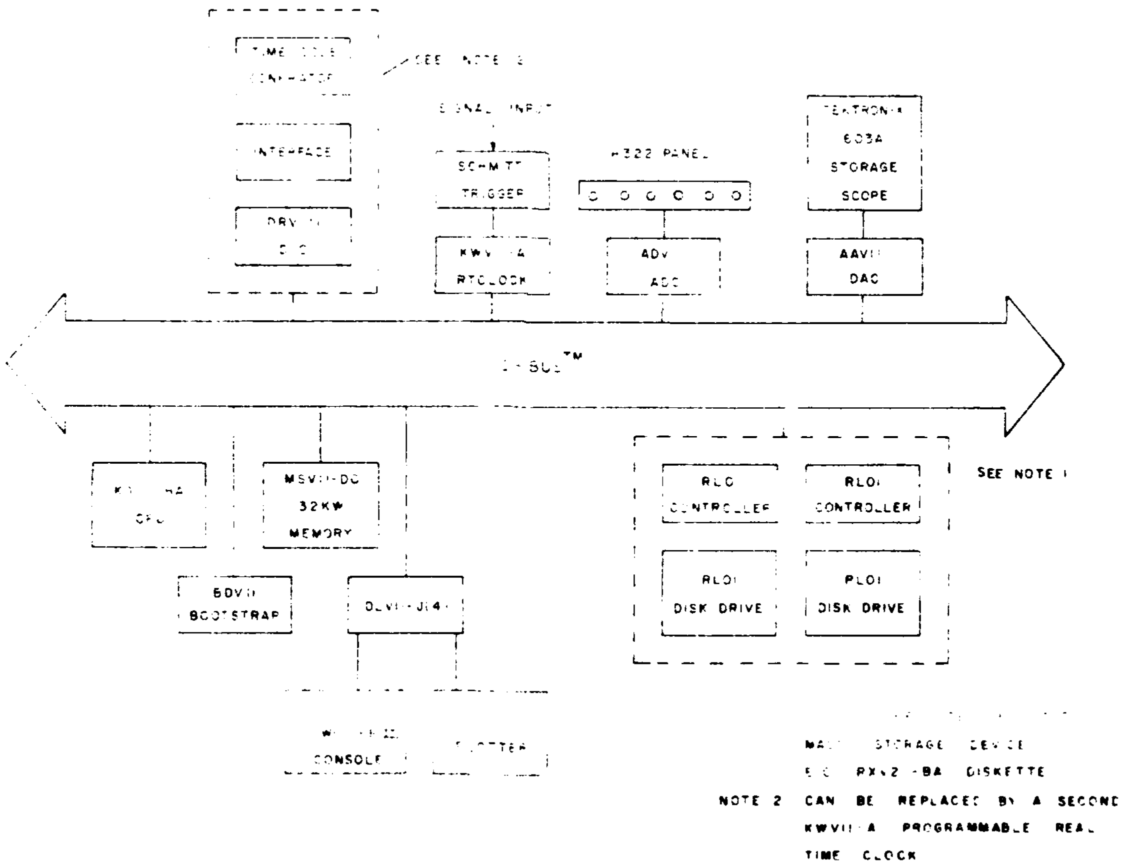


Figure 8.2 PDP 11/03-based Data Reduction System

The PDP 11/03 system is configured with 32K words of MOS memory. The processor also has four DLV11-J asynchronous serial interfaces to provide RS-232C interface to four peripheral devices, including the LS-120 console terminal. In the proposed system, a Hewlett-Packard 7221A plotter is included to provide hard copy outputs. A dual RL01 hard disk system or dual RXV-21 double density floppy disk system is used for mass storage of data and programs. The RL01 hard disk can provide 10 million bytes of on-line storage and a much faster data transfer rate. The RXV11-21 floppy disk system, on the other hand, can only provide 1 million bytes of on-line storage, but the low cost of floppy disks makes it an attractive alternative.

The LPS is not supported by the LSI-11 processor but a number of modules are available which can perform the functions of the LPS. They are listed below:

- KWV11 programmable real time clock;
- ADV11 16 channel, 12 bit A/D converter;
- AAV11 4 channel, 12 bit D/A converter;
- DRV11 Digital I/O interface.

The hardware descriptions for these modules and other necessary peripherals are listed in Table 8.1.

The Tektronix 603 storage scope is used for viewing and scaling the waveforms. The AAV11 D/A converter drives the storage scope. Both the storage scope

KD11-HA	LSI-11/2 Central Processor Unit with power fail/auto restart. LSI-11 bus interface and vector in-[illegible] handling. Single board, size: 8.9 in. x 5.2 in. (22.8cm x 13.2cm).
MSV11-DD	32K byte Random Access Memory (RAM) including on-board refresh. Single board, size: 8.9 in. x 5.2 in. (22.8cm x 13.2cm).
RLV11-AK	A 5.2 M byte, top loading removable cartridge subsystem for all 11/03L LSI-11 systems, two quad control boards, size: 8.9 in. x 10 in. (22.8cm x 25.4cm). NOTE: Requires H9273 or BA11N backplane.
RL01-AK	Add on 5.2 M byte top loading removable cartridge disk drive.
RXV21-BA *RXV21-BD*	Dual drive double density, 1024K-bytes capacity plus DMA LSI-11 interface board. Single board, size: 8.9 in. x 5.2 in. (22.8cm x 13.2cm).
DLV11-J	Four independently configurable serial line units. Supports RS-422 and RS-423 (compatible with RS-232 C). Selectable parity, data and stop bits. Baud rates from 150 to 33400. Single board, size: 8.9 in. x 5.2 in. (22.8cm x 13.2cm). NOTE: Requires one cable per line and H3270 test connectors for diagnostics.
LA120-BA	DECwriter III keyboard printer with numeric pad which *operates* on RS-232C standard. 132-column, 7 x 7 dot matrix, 180-cps smart bi-directional printing designed for serial 1200 baud communications. *20mA current loop interface optional.*
AAV11-A	[illegible], 4-channel (buffered), D/A converter. Output ranges: ±2.56V, ±5.12V, ±10.24V, 0-5.12V, 0-10.24V. Single board, size: 8.9 in. x 10 in. (22.8cm x 25.4cm).
ADV11-A	12-bit, 16-channel, single-ended, (8-channel differential) A/D converter. Input ranges: ±5.12V. [illegible] hold circuitry. 50μs conversion time. Single board, size: 8.9 in. x 10 in. (22.8cm x [illegible]).
KWV11-A	Programmable Crystal Clock with frequencies from 100 Hz to 1 MHz plus 60 cycle and external input. Single board, size: 8.9 in. x 10 in. (22.8cm x 25.4cm).
DRV11	Parallel Line Interface Unit. 16-bit diode-clamped input; 16-bit latched-drive output. Protocol and control signals. Suggested cables BC07D or BC03R. BC08R can be used as a diagnostic cable. Single board, size: 8.9 in. x 5.2 in. (22.8cm x 13.2cm).
BA11-[illegible] *BA11-[illegible]*	Expander Box. Includes 4 x 9 backplane (LSI-11 bus on slots A & B only) with 240 WATT power supply.

RT-11 [illegible]-Time Operating System designed for the single user. Includes single job monitor and foreground/background monitor. RT-11 system programs include EDIT, MACRO-11, LINKER, ODT, and utilities. Minimum hardware: CPU with 16K byte (32K byte for foreground/background operation).

Fort[illegible]-11 [illegible]nsion Standard Fortran with symbolic debugger usable on LSI Bus products, also includes a library of [illegible] subroutines not usable on LSI Bus products. Prerequisite: license to use RT-11.

Table 8.1 LSI-11 System Supporting Hardware Description

and the D/A converter have 12-bit resolution (4096) so they interface nicely. The Hewlett-Packard plotter, with the proper software configuration, can draw waveform images on to a hard copy in addition to plotting the reduced conductivity data. This would permit the detailed study of waveforms and probe response characteristics at a higher resolution than what is presently possible. The Schmitt-trigger available on the LPS can very easily be built using op amps to trigger the KWV11 programmable real-time clock. If a second KWV11 is available, it can be used in place of the DRV11 digital I/O and the time code generator. It can be programmed to run at any desired frequency for timing information. For example, 10 Hz will give 0.1 second resolution; 100 Hz will give 0.01 second resolution in timing. In order to prevent overflow from the 16 bit word, the time count can be stored in 2 words using the "ADD" and "ADC" instructions. In this case, for example, up to 42949673 seconds can be stored with 0.01 second accuracy[11].

The PDP 11/03 system has the RT-11 V3B operating system and FORTRAN IV software support. Plotting subroutines written in FORTRAN are also available from Hewlett-Packard. Therefore, this system could be operating in a very short period of time with minimum software

[11] If the clock frequency = 100 Hz, i.e. each second has 100 counts, then 32 bits provides $2^{32}-1$ counts; therefore time = $(2^{32} - 1)/100$ = 42949673 seconds.

modifications to our existing system.

8.2 COMPONENT APPROACH

Due to the enormous number of micro-processor chips available, this discussion will not attempt to focus on any particular chip. Instead, a general configuration will be outlined so that the system designer can use it as a guide line for the particular design (Figure 8.3).

The heart of the system is the data acquisition phase as described in this paragraph. In the component approach, a fixed frequency clock is fed into a binary counter for frequency measurement. When a pulse is detected from the incoming signal, the Schmitt-trigger is fired causing the contents of the counter to be latched into the tri-state latch. It also sets the flip-flop that is tied to the interrupt request line (INTR) of the micro-processor chip. (A flip-flop is used because most micro-processors will only sample the interrupt request line after the last machine cycle of each instruction. If the flip-flop were not used, the interrupt request might be lost [Liu (1979)].) After the (INTR) signal is recognized by the micro-processor, an interrupt acknowledge (INTA) signal will be issued to reset the flip-flop. Depending on the hardware and/or software configuration of the particular micro-processor system, con-

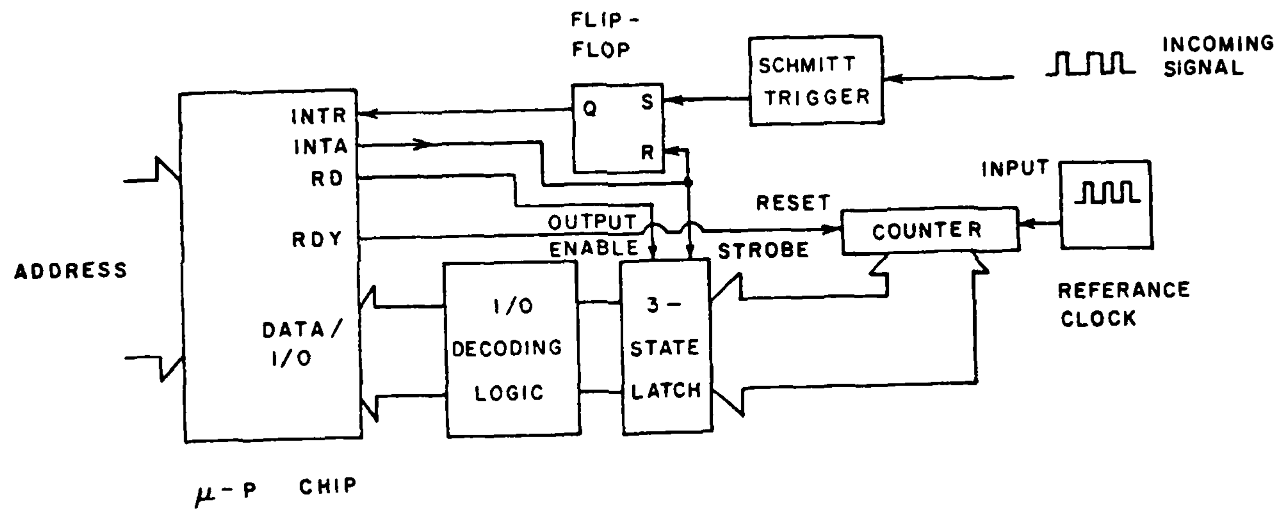


Figure 8.3 Micro-processor-based Data Acquisition System

trol is passed in someway to an interrupt service routine. The interrupt service routine will issue an I/O read to read the content of the counter. This, in turn, causes the read (RD) line to activate the output enable (OE) line on the tri-state latch which puts the count on the data bus of the micro-processor system. The count can therefore be read in from an I/O port of the micro-processor. (The address of the port is determined by the I/O decoding logic.) After the interrupt service routine finishes its task, control returns to the normal program sequence and a ready (RDY) signal will be issued. This signal resets the counter which is ready for the next pulse.

After the data are digitized, they need to be stored and reduced using either a micro-processor development system or a mini-computer.

8.3 COMPARISON BETWEEN THE TWO APPROACHES

The system approach is, of course, more costly. Yet the reliability and maintenance available from OEM products impose a much greater advantage over the component approach. In addition, the compatibility of the LSI-11 and PDP 11 computer systems allows minimum modifications to the existing software.

The component approach requires a tremendous amount of design effort but with less hardware cost.

Therefore, it can also be an attractive alternative for a production environment.

CHAPTER IX

ERROR ANALYSIS

In this chapter, errors that might possibly develop during the different stages in the data reduction process are considered and their significances to the reduction results are analyzed. Improvements to the present system are proposed.

9.1 ERRORS RESULTING FROM A FIXED RATE CLOCK AND A FIXED WORD LENGTH

Two kinds of errors result from using a fixed frequency clock reference on a fixed word length computer. The first one is very obvious. Since the clock is counting at 100 kHz, any incoming signal with a frequency greater than 100 kHz is not detected. The normal operating frequency of the probe is between 0 and 200 Hz only and thus the significance of this kind of error is very small.

The other kind of error results from the digital computer having a fixed word length. In the case of the PDP 11's, the word length is 16 bits. This means that an unsigned integer representation can at most be $2^{16}-1$, i.e. 65535. In other words, when the clock is counting at 100 kHz, only 65535 counts can be recorded before overflow occurs. This number corrosponds to 0.65535 second or a frequency of 1.53 Hz. In normal operation, a

frequency lower than 1.53 Hz is unlikely but this situation is possible if there is signal dropout. In this case, the clock counts up to 65535 counts, and then any count beyond that will reset the counter to zero and it starts counting from zero again. In effect, 0.65535 second is lost and the count at the end of this pulse is not correct. This was a major problem in the past and there was no easy solution because of the chosen design model. In the present system, however, the problem is insignificant because the introduction of the external time code generator has completely separated the programmable real-time clock from the task of keeping the in-flight timing. Whenever dropout occurs, only one bad point is introduced. The timing distortion associated with this point is not carried beyond that particular waveform because each time the push button is pressed, Pass 1 will go out to the external time code generator to get timing information which re-synchronizes itself.

When signal dropouts occur during the data acquisition phase, the operator can allow the dropouts to elapse until normal signal playback recovers. Then the push button switch can be issued and timing is re-synchronized. During the data reduction pass, the operator will notice that the waveform is shortened due to the data loss during dropout. The waveform can be discarded and reduction can resume on the next waveform.

9.2 ERRORS RESULTING FROM I/O ACTIVITIES

Pass 1 is the time-critical phase since it has to capture the incoming signal in real time. Looking at the data acquisition method discussed before, the time-critical portion of Pass 1 is the interrupt service routine. When the Schmitt-trigger is fired by an incoming pulse, an interrupt is generated. Control is then passed to the interrupt service routine and during this period, further interrupts are disabled. If another pulse comes in during this time, it is not recognized. Coding of the interrupt service routine has been carefully optimized to reduce execution time (see Appendix D.1). With our particular hardware and software configuration, under normal circumstances, each pulse takes 125.5 us[12] to be stored. The corresponding bandwidth is 7.95 kHz (including the A/D converter sampling time). In the worst case for which the buffer has been filled and I/O activity is required, 215.9 us are needed, thus giving a bandwidth of 6.35 kHz. With the normal operating range of the probe frequencies (0 to 200 Hz), this error is insignificant.

The above calculations are based on the fact that the RT-11 operating system has the full support of asynchronous I/O. Each time I/O activity is needed, the program only issues the I/O request through the use of "sys-

[12] 1 us = 10^{-6} second.

tem macros". Control then returns to the calling program after the I/O request has been recognized. The I/O activities will be executed in the background asynchronously [RT-11 System Reference Manual (1975)].

9.3 ERRORS RESULTING FROM DISPLAY AND CURVE FITTING

In the field of computer graphics, there has always been the problem of mapping the world coordinates into the device coordinates since the world coordinate system has almost unlimited addressable space whereas the device coordinate system is always restricted by the limited discrete representation of physical addressable spaces. The same problem arises here since the count from the programmable real-time clock can range from 1 to 65535, i.e. there are 65536 possible readings. However, there are only 4096 addressable points on the storage scope, thus implying that points with close values will fall into the same spot on the screen. Since the resolution of the human eye is far less than the scope's resolution, this effect is not significant.

The next problem arises from the basic relationship of calculating frequency from count. These two parameters are inversely proportional to each other and thus a hyperbolic relation is expected. In contrast to a linear relationship between count and frequency, the frequencies in the middle portion, for example from 60 to

160 Hz, will only be resolved by 1,041 counts[13]. This means a 1-in-10 resolution instead of 1-in-100,000 as expected. This in part is the reason for choosing such a high reference frequency as 100 kHz.

The other type of error comes from the least square curve fitting program. The algorithm for slope fitting is designed such that a weighting factor between 0.0 and 1.0 is assigned to each point according to how far that point is from the first fitted line. The implication is that points farther away from the fitted line will bear less significance to the final fit. However, there are two important points to remember. First, the choice of the time segment to be scaled, i.e. the two end points, is very important. If the end points are not correct, a correct result is less likely to occur. Secondly, if there exists an overwhelming amount of noisy data in the region between the two end points, the resultant slope will not be correct. This is largely because the first trial will assign an equal weight to every data point. If the first trial is not correct, as in the case of a lot of noise, the successive trials will not possibly be correct. To solve this problem, a small clean portion of the waveform needs to be chosen to generate the fit. This usually achieves a very satisfactory result.

[13] if cnt1 = 100kHz/60 = 1666; cnt2 = 100kHz/160 = 625; then (cnt1 - cnt2) = 1000.

Finally, the reduction of waveforms having very steep slopes (large conductivity values) requires the expansion routine to blow up a very small portion of the waveform on the screen. This allows pin-pointing to the exact end points and in turn, scaling of the proper time segment of the waveform.

9.4 ERROR RESULTING FROM THE TIME CODE GENERATOR

For the present hardware available, the time code generator is clocked every second. Therefore, timing can be at most one second off from the actual value. There are two ways to improve this accuracy. The first method requires a time code generator that gives more resolution in output timing. The second method is to put in a correction factor to include the timing elapsed up to the interval of the designated slope. This can very simply be done by summing up all the counts to the middle point of the time interval scaled. By storing this corrected time factor in the extra space of the header, the timing accuracy is improved. This would be especially important during the early stage of the flight where the payload is decending at a relatively high velocity.

CHAPTER X

RESULTS AND FUTURE IMPROVEMENTS

In this chapter, electrical conductivity values obtained from the computerized reduction system are presented. In addition, a section is presented describing some of the problems possibly encountered during the reduction process and how they might be corrected. Finally, possible hardware and software improvements are discussed.

10.1 RESULTS

The end results of this computerized reduction scheme are the electrical conductivity values measured in the middle atmosphere. Using this reduction procedure, conductivity data from four blunt probes recently launched on super Loki rockets are shown in Figures 10.1 to 10.4. All four rocket launches were conducted at relatively high latitudes. The two 1977 rocket launches occurred at Poker Flat, Alaska (65° N, 147° W) as part of a program to study the ionization effects of auroral energetics on the middle atmosphere. The two 1979 launches were part of a solar eclipse rocket program conducted at Red Lake, Ontario, Canada (51° N, 94° W).

In all of the figures, the plus and minus signs represent positive and negative conductivity values, respectively, while the dots indicate where the measure-

ments are comparable in value. The daytime measurements (Figure 10.1, 10.2 and 10.4) at higher altitudes show noticed differences in the polar conductivity components for the same altitude, with the negative conductivity values being relatively larger. The larger negative conductivity values are thought to demonstrate the presence of electrons, which have relatively larger mobility values. In general, more variability is observed in the reduced negative conductivity measurements, which is indicative of the difficulty in reducing the electron current component.

The nighttime conductivity measurements (Figure 10.3) generally show little differences in the polar conductivity components at the same altitude, which is expected when there are no electrons present. Surprisingly, the negative conductivity values at lower altitudes are relatively smaller than the corresponding positive values, which is generally not the observed case. These relatively smaller negative conductivity values are probably attributed to smaller ion mobility values.

10.2 USER'S EXPERIENCE

The advantages of this data reduction package are enhancement in speed, greater flexibility and improved accuracy.

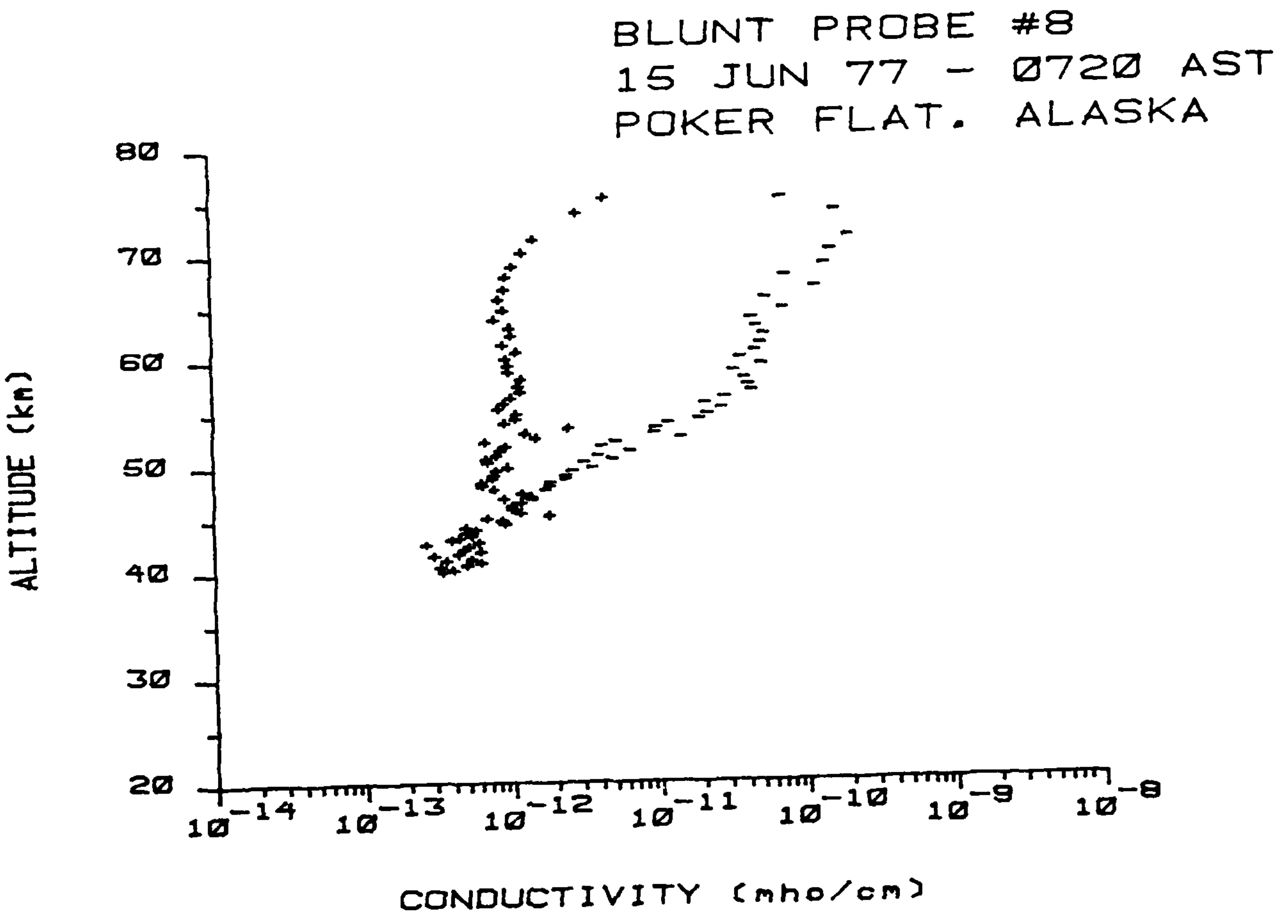


Figure 10.1 Electrical Conductivity Measurements for June 15, 1977 at 0720 AST

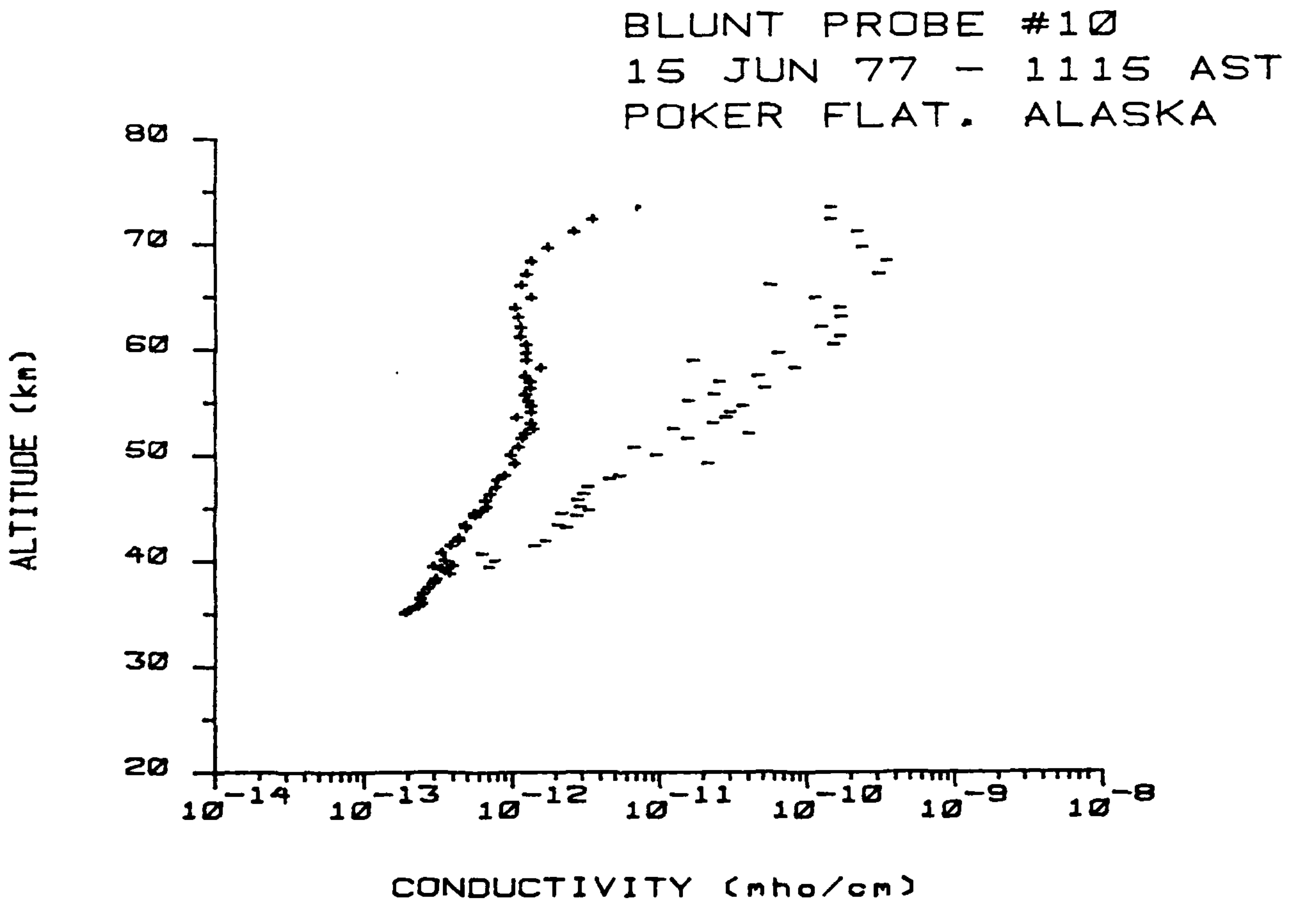


Figure 10.2 Electrical Conductivity Measurements for June 15, 1977 at 1115 AST

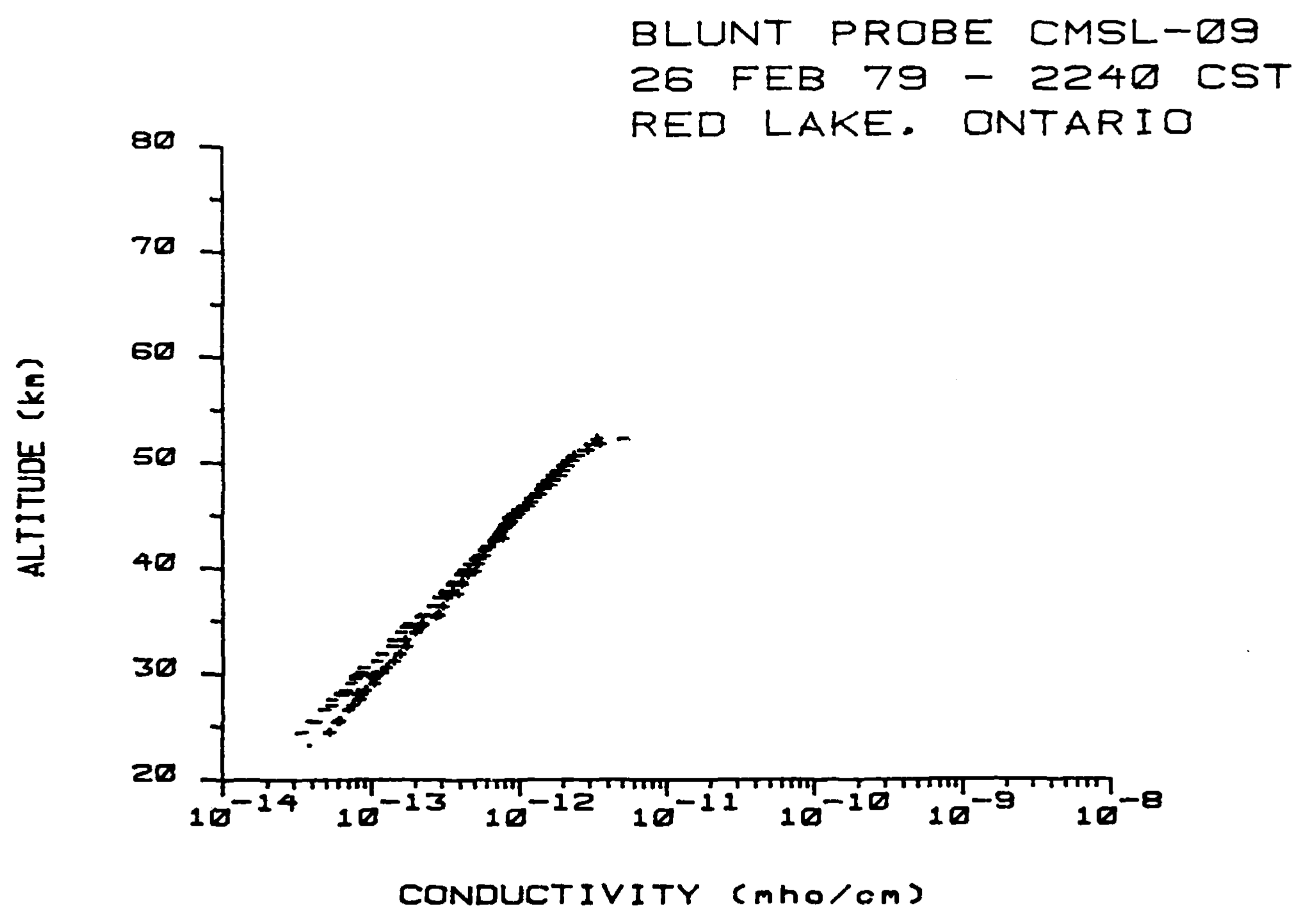


Figure 10.3 Electrical Conductivity Measurements for February 26, 1979 at 2240 CST

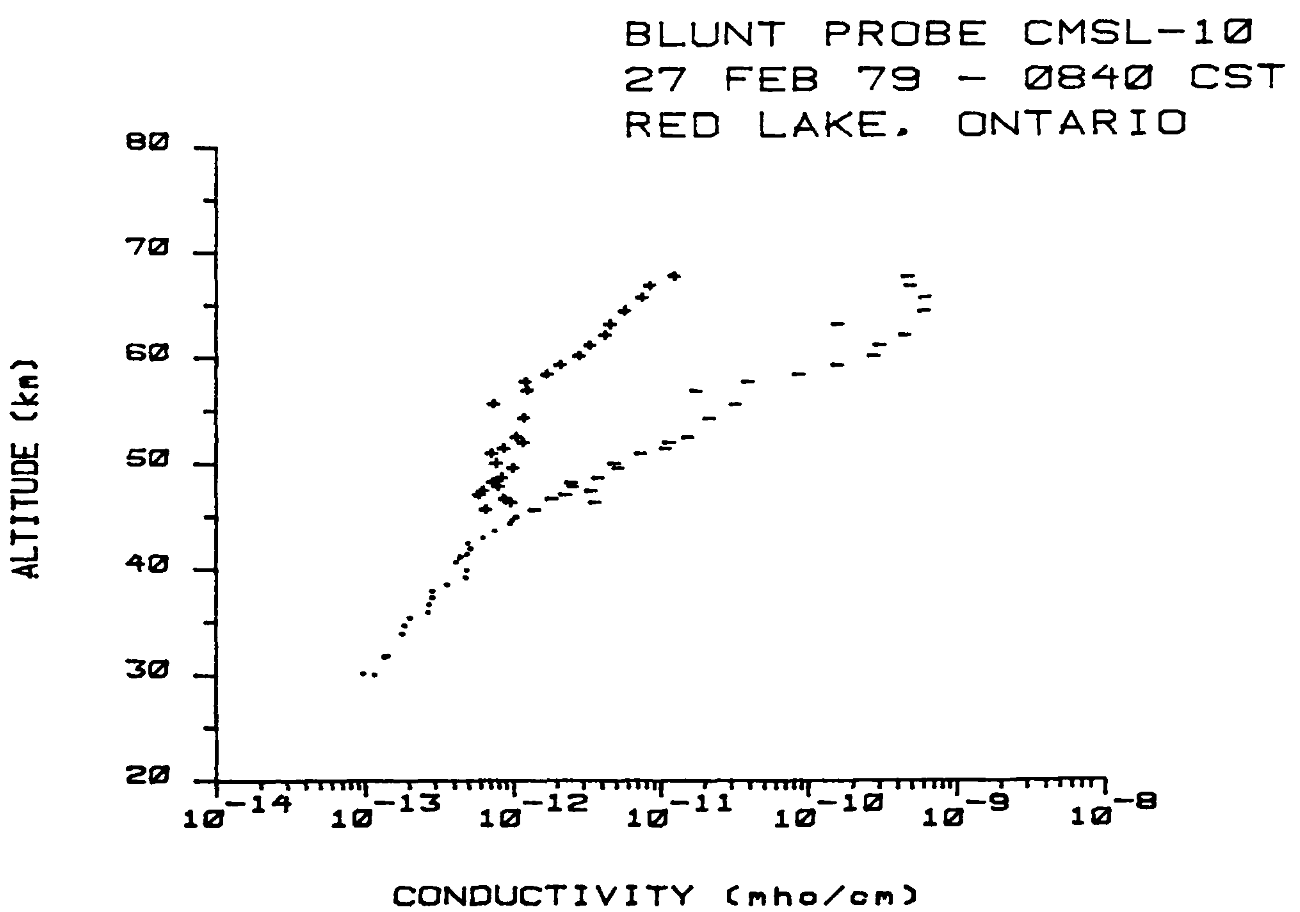


Figure 10.4 Electrical Conductivity Measurements for February 27, 1979 at 0840 CST

Digitization only needs to be done once for each flight and therefore a lot of play-back time is saved. This system frees the user from tedious manual scaling of waveforms. Scaling (by a numerical method), line generation, calculation and data storage are all done automatically. Furthermore, the results are all self-contained in the header for each waveform of each flight. This information is in machine readable form and can be utilized directly by machines. For example, a plotting utility program has been written to read the results from the headers and plot them out on the Tektronix 603 storage scope for viewing. Another routine has been written to extract data from the header and plot the results on the hard copy plotter of the PDP 11/45. Different outputs are thus generated according to the format required without any complications. Different sizes of plots, different titling and labels are all done easily.

Each 45-minute flight occupies approxmately 1000 blocks of disk space. If storage is tight, the Schmitt-trigger level on the incoming signal can be turned off periodically during the latter part of the flight where the altitude changes slowly. Since the external timer runs asynchronously, timing is still kept correctly.

Accuracy is another reason for using the reduction package. The data digitization method enhances the resolution and in addition, the expansion routines can

blow up any portion of the waveform so that detailed study of the waveform's characteristics are possible. Using numerical techniques to compute the slope are also advantageous, especially for very steep or very flat slopes.

Finally, sporadic noise pulses or telemetry drop-outs will often corrupt data waveforms displayed on a strip chart such that they are not suitable for scaling. It has been observed that these effects are often reduced using computer techniques such that conductivity information can still be extracted from the waveform.

10.3 FUTURE IMPROVEMENTS

In this section, future improvements to the data reduction system are discussed. The improvements include steps toward a more automated reduction system, more sophisticated hardware and more fully coordinated software.

10.3.1 TOWARDS AUTOMATION

Probably the most significant step toward automation of the reduction process actually involves a probe redesign such that an indication of the time for zero-volt probe potential is also telemetered with the current-voltage response. Such an indication would permit adaptation of a new computerized waveform segmentation scheme to replace the present operator-controlled

procedure. In addition, it would help in identifying the two separate polar currents collected by the probe. Finally, it could be used to estimate probe potential during other times of the waveform, which is particually relevant in the further reduction of probe data to estimate electron density (for blunt probes) and ion density (for Gerdien condensers).

10.3.2 HARDWARE IMPROVEMENTS

More sophisticated hardware will of course, make the data reduction system more attractive. A display processor equipped with a light pen could replace the storage scope, thus allowing a more convenient way of scaling waveforms. A plotter interfaced to the system not only would provide a final form of output but with a proper software setup, would draw each scaled waveform on paper as well. This would provide an opportunity for studying the detailed structure of the probe's current response characteristics. Lastly, a second programmable real-time clock could be used for obtaining accurate timing information under software control.

10.3.3 SOFTWARE IMPROVEMENTS

With the digitized data entered into the computer, future software improvements are concerned with further manipulation of this stored information. Specifically, further software development might include: im-

provement of the method for accumulating time; estimation of probe potential; calculation of electron number density; refinement of plotting routines to include multi-images on each output; and expansion of library routines.

REFERENCES

Conley, T.D., Mesospheric positive ion concentration mobility and loss rates obtained from rocket-borne Gerdien condenser measurments, Radio Sci. 9, 575, 1974.

Croskey, C.L., In situ measurments of the mesospheric and stratosphere, Scientific Report No. 442, Ionospheric Research Laboratory, The Pennsylvania State University, 1976.

Croskey, C.L., L.C.Hale and S.F.Lieden, Results of ionization measurments in the middle atmosphere, Space Res. XVII, in press, 1977.

Dahl, O.-J, E.W. Dijkstra and C.A.R. Hoare, Structured Programming, Academic Press, New York, 1972.

Domagalski, K., "Gerdien condenser instrumentation for measuring high-latitude middle atmosphere electrical parameters", Master's thesis, The University of Texas at El Paso, El Paso, Texas, 1979.

FORTRAN/RT-11 Extansion Manual, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

Hale, L.C., Parameters of the low ionosphere at night deduced from parachute brone blunt probe measurments, Space Res. VII, North-Holland, Amsterdam, 1975.

Hale, L.C. and D.P. Hoult, A subsonic D-region probe theory and instrumentation, Scientic Report No. 247, Ionospheric Research Laboratory, The Pennsylvania State University, 1965.

Hale, L.C., D.P. Hoult and D.C. Baker, A summary of blunt probe theory and experimental results, Space Res. VIII, North-Holland, Amsterdam, 320, 1968.

Kernighan, B.W. and P.J. Plauger, Software Tools, Addison-Wesley, Reading, Massachusetts, 1976.

Liu, Y.C., Private communication, 1979.

LPS 11 Laboratory Peripheral System Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, 1973.

Microcomputer Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1976.

Mitchell, J.D., R.S. Sagar and R.O. Olsen, Positive ions in the mddle atmosphere during sunrise conditions, Space Res. XVII, 199, 1977.

602/603 Monitor Instruction Manual, Tektronix, Inc., Beaverton, Oregon, 1971.

Newman, W.M. and R.F. Sproull, Principles of Interactive Computer Graphics, 2nd Edition, McGraw-Hill, New York, 1979.

PDP 11 FORTRAN Language Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, 1974.

PDP 11/04/05/10/35/40/45 Processor Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

PDP 11 Peripherals Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

Pedersen, A., Measurment of ion concentrations in the D-region of the ionosphere with a Gerdien condenser rocket probe, FOA 3 Report A607, Research Institute of National Defense, Stockholm, Sweden, 1964.

Rose, G., and H.U. Widdel, Results of concentration and mobility measurments for positively and negatively charged particles taken between 85 and 22 km in sounding rocket experiments, Radio Sci. 7, 81, 1972.

RT-11 System Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, 1975.

Shih, S.W., "A computerized system for the reduction of the middle atmospheric electrical conductivity data", Master's thesis, The University of Texas at El Paso, El Paso, Texas, 1977.

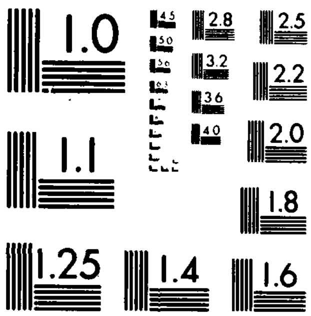
"Status Report of the Graphics Planning Committee of ACM/SIGGRAPH", Computer Graphics, Vol. 11, Number 3, Association of Computation Machinary, New York, 1977.

Technical Manual, Model 1706 and 1710 Time Code Generator, Eldorado Electrodata Corporation, Concord, Califonia, 1970.

AD-A088 734 TEXAS UNIV AT EL PASO SCHELLENGER RESEARCH LABS F/G 4/1
THE DESIGN AND IMPLEMENTATION OF A COMPUTERIZED DATA REDUCTION --ETC(U)
JAN 80 K J HO, J D MITCHELL DAAD07-78-C-0010
UNCLASSIFIED SR3-80-UA-77 NL

2 of 2
AD A088 734

END
DATE FILMED
10 80
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNIX Programmer's Manual, 6th Edition, Bell Telephone Laboratory, Murray Hill, New Jersey, 1975.

Wirth, N., Algotithm + Data Structure = Program, Prentice-Hall, Englewood Cliff, New Jersey, 1976.

APPENDIX A

PROGRAMS USED IN THE DATA REDUCTION SYSTEM

A.1 PROGRAMS UNDER THE RT-11 SYSTEM

PASS1 - Data acquisition

PASS2 - Data segmentation and header insertion

PASS3 - Data display and reduction

PLOT - Plotting conductivity values vs altitude on the Tektronix 603 storage scope

XTRACT - Extract conductivity values from the header and put them into Basic-Plus virtual array format for plotting on the PDP 11/45

PIP - DEC's system program for printing or transfering files between peripheral devices

A.2 PROGRAMS UNDER THE UNIX SYSTEM

rtpip - Transfering files between the UNIX and RT-11 systems

plot - General purpose plotting routine

APPENDIX B

USER'S MANUAL

B.1 HARDWARE SETUP FOR PASS 1

[i] See the schematic in Figure 4.1.

[ii] Connect the output of the playback tape recorder to the input of the Schmitt-trigger 2 on the LPS.

[iii] Connect the pulse generator to channel 0 of the A/D converter on the LPS.

[iv] Connect the terminating signal output to channel 2 of the A/D converter on the LPS.

[v] If there is another data signal, connect it to channel 1 of the A/D converter.

[vi] A strip chart machine can optionally be connected to the output of the Schmitt-trigger 2 to obtain a hard copy of the data waveform.

[vii] An oscilloscope can optionally be connected to both the input and the output of the Schmitt-trigger 2 for monitoring.

B.2 OPERATING PROCEDURE FOR PASS 1

[i] Set up the hardware configuration as described in Appendix B.1.

[ii] Turn on all the equipment.

[iii] Play back portions of the tape for a few times to determine the correct triggering level on the Schmitt-trigger 2.

[iv] Reset the time code generator.

[v] Boot up RT-11 and execute the program "PASS1.SAV".

[vi] Erase the storage scope.

[vii] Start the tape recorder. The storage scope will indicate the data are entered.

[viii] Start the time code generator when launch is detected (either from the scope or the strip chart).

[ix] Push the switch on the pulse generator whenever a new waveform is recognized.

[x] Issue the terminating signal (a 5 V falling pulse) when the data end.

[xi] Execute the program "PIP" to see if the file "PASS1.TMP" is present.

B.3 OPERATING PROCEDURE IN PASS 2

[i] Obtain data from Pass 1 as described in Appendix B.2.

[ii] Boot up RT-11.

[iii] Execute the program "PASS2.SAV".

[iv] Answer all the questions the program asks concerning the probe's information.

[v] The program will execute automatically. The LED display on the LPS will show the time of each waveform sequentially.

[vi] When the program terminates, the file "PASS1.TMP" can be deleted using "PIP".

[vii] Use "PIP" to check if "PASS2.TMP" is present.

B.4 OPERATING PROCEDURE IN PASS 3

[i] Turn on the storage scope and erase it.

[ii] Boot up RT-11 and execute "PIP" to find out the size of the file "PASS2.TMP".

[iii] Execute "PASS3.SAV".

[iv] Answer the questions requested by the program:
  a) The name of the altitude file is an ASCII file that contains the time vs altitude radar data.
  b) The number of blocks in "PASS2.TMP" is the size obtained in step [ii] above.
  c) The calculation constant is:

$$\frac{R}{2r^2 R_{CAL}(df/dt)_{CAL}} \quad \text{FOR BLUNT PROBE}$$

$$\frac{\ell n\,(r_o/r_i)}{2\pi\,\ell\,R_{CAL}\,(df/dt)_{CAL}}$$ FOR GERDIEN CONDENSER

1.0 FOR CALIBRATION

for conductivity calculations.

[v] The program proceeds and asks from which waveform number to begin reduction. Reduction can start at any waveform desired. A <CR> implies the first.

[vi] The complete waveform chosen will appear on the screen with a vertical scale from 0 to 200 Hz. The time for that waveform will appear on the LED display as seconds into the flight.

[vii] The program will ask if expansion in the X and/or Y direction(s) is(are) desired. If no expansion is needed, go to step [ix].

[viii] The program will request the X and Y boundaries for expansion. A <CR> alone means display the whole waveform from 0 to 200 Hz again. The boundary supplied in the X direction means only displaying in the selected X-direction limit but from 0 to 200 Hz vertically. If both the X and the Y directions are given, the display will be on the selected portion in both directions. The program will then go back to step [vii] and repeat until no more expansions are desired.

[ix] The program will ask for the two end points on the positive conductivity slope. If a <CR> is given, go to step [xi].

[x] The program receives the two end points and will find the best-fitted straight line betweem them. Then a solid line will be generated on the screen. The program will ask for break points again. If the fit is satisfactory, type a <CR>, otherwise input another two points and step [x] will repeat.

[xi] If step [x] has been executed, that means a positive conductivity value is present; go to step [xii], otherwise go to step [xiv].

[xii] The program will execute step [vii] through [xi] again for negative conductivity calculations. If a <CR> is given, that means the positive and negative conductivities are equal.

[xiii] The program will ask for another set of positive and negative conductivities. If they are present, steps [vii] through [xiii] will repeat.

[xiv] The reduced conductivities, altitude and time will be printed on the console.

[xv] The program will ask to continue or skip forward. A "C" means continue to the next waveform; a "S" means to skip some of them. (The program will ask how many waveforms are to be skipped.) A <CR> means to terminate the process.

[xvi] The process will be repeated until termination is issued from step [xv] or until the end of data is encountered.

[xvii] An ASCII file called "PASS3.TMP" with all the reduced information has been created. It can be examined or printed.

B.5 LIBRARY CREATION

[i] A data library can be created by using PIP.

[ii] The following convention for extensions on data files has been employed to distinguish between different file contents:

```
*.CAL calibration waveform file (PASS2 format)
*.DAT in-flight data file  (PASS2 format)
*.LST results    (ASCII)
*.POS positive conductivity vs
         altitude file   (virtual array)
*.NEG negative   (virtual array)
*.COM positive = negative  (virtual array)
*.ALT time vs altitude radar data (ASCII)
```

APPENDIX C

PROGRAM LINKAGE

The following commands give the program modules' names and linkage procedure for producing the proper executable programs under the RT-11 Operating System. They properly compile and assemble the main programs and subroutines into object modules and use the RT-11 linker (LINK) to produce executable programs as follows:

```
PASS1.SAV<PASS1.OBJ
PASS2.SAV<MAIN2.OBJ,PASS2.OBJ,QUEST.OBJ/F
PASS3.SAV<PASS3.OBJ,SUB1.OBJ,SUB2.OBJ,SUB4.OBJ,SLOPE.OBJ/F
PLOT.SAV<PLOT.OBJ/F
XTRACT.SAV<XTRACT.OBJ,SUB1.OBJ/F
```

APPENDIX D: PROGRAM LISTINGS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       PASS1 OF DATA PROCESSING                    ;
;                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THIS ROUTINE USES THE LPSKW TO COUNT
;       AT 100 KHZ AND THE SCHMITT-TRIGGER-2 TO
;       COUNT INCOMING PULSE FREQUENCY
;       INPUTTED FROM THE TAPE RECORDER
;
;
;       THE ADCS PROVIDE COMMANDS FOR THE ROUTINE
;       AS FOLLOWS:
;
;       CHANNEL 0 -- A +5V PULSE OF 50 MS IS
;       USED FOR WAVEFORM SEGMENTATION
;       TIMING INFORMATION WILL BE SAMPLED
;       FROM THE DIO PORTS OF THE LPSDR
;
;       CHANNEL 1 -- NOT USED AT PRESENT BUT IS
;       RESERVED FOR FUTURE EXPANSION
;
;       CHANNEL 2 -- TERMINATION SIGNAL
;       NORMAL HIGH (+5V)
;
;
;
;       SOME CONSTANT DEFINITIONS
;
;       TO USE HIGHLY PARAMETERIZED PROGRAMMING
;       CAN REDUCE CHANCE FOR MAKING ERROR WHEN
;       THE PROGRAM HAS TO ADAPT TO OTHER SYSTEM
;
;
        BUFSIZ = 4096.          ; BUFFER SIZE = 4K
        FILSIZ = 1000.          ; DEFAULT FILE SIZE = 1000 BLOCKS
        RECLEN = 16.            ; 4K BUFFER = 16 BLOCKS LONG
        CHANO = 1               ; BIT PATTERN TO START A ADC IN CHO
        CHAN1 = 401             ; START ADC AT CHANNEL 1
        CHAN2 = 1001            ; START ADC AT CHANNEL 2
        KWENBL = 1505           ; START LPSKW, 100KHZ, REPEAT MODE
        ERASE = 10000           ; ERASE THE SCOPE
        VCRDY = 2012            ; LPSVC Y-MODE, FAST INTENSIFY
        EOWF = 0                ; END OF WAVEFORM MARK
        NOINT = 340             ; BR7, NO OTHER INTERRUPT
        ZEROV = 5000            ; DIGITAL VALUE OF 0 V FOR ADC
        YVAL = 4000             ; MIDDLE OF SCREEN
        ICNT1 = 4000.           ; COUNT FOR IDEL LOOP1
        ICNT2 = 4000.           ; COUNT FOR IDEL LOOP2
        XINC = 2                ; DISTANCE BETWEEN POINTS
;
;
;
;
;
;       LPS ADDRESSES DEFINITION
;       REFER TO LPS OPERATING MANUAL FOR
;       FURTHER REFERENCES
;
;
        .NLIST
        ADST = 170400           ; LPSAD STATUS
        ADBF = 170402           ; LPSAD BUFFER
        KWST = 170404           ; LPSKW STATUS
```

```
                KWBP = 170406           ; LPSKW BUFFER/PRESET
                DRST = 170410           ; DIGITAL I/O STATUS
                DRIN = 170412           ; DIGITAL I/O INPUT
                DROUT = 170414          ; DIGITAL I/O OUTPUT
                VCST = 170416           ; LPSVC STATUS
                VCX = 170420            ; LPSVC X
                VCY = 170422            ; LPSVC Y
                KWIV = 324              ; LPSKW INTERRUPT VECTOR
                                        ; NOTE THIS IS NON-STANDARD
        ;
        ;
        ;
        ;
        ;       SOME MACRO DEFINIATIONS
        ;
        ;       JUMP TO SUBROUTINE THRU PC
        .MACRO  CALL    A
        JSR     %7,A
        .ENDM
        ;       RETURN FROM SUBROUTINE
        .MACRO  RETURN
        RTS     %7
        .ENDM
        ;       WAIT IDLE UNTIL READY BIT (BIT 7) IS SET
        ;       A DEC'S CONVENTION
        .MACRO  WAIT    A,?B
        ;       A IS THE DEVICE ADDRESS, B IS AN
        ;       ASSEMBLER GENERATED LABEL
B:      TSTB    @#A
        BPL     B
        .ENDM
        ;
        ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;                                                ;
        ;               MAIN PROGRAM                     ;
        ;                                                ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;
        ;
        .TITLE PASS1
        .SBTTL  PASS1 OF DATA PROCESSING
        ; MAIN PROGRAM CONTROL SECTION
        .CSECT  MAIN
        ; ENTRY POINT NAME IS PASS1
        .GLOBL  PASS1
        ; CALL FOR SYSTEM MACROS
        .MCALL  ..V2..,.REGDEF
        .MCALL  .PRINT,.FETCH,.EXIT
        .MCALL  .ENTER,.CLOSE,.WRITE
        .MCALL  .WRITW,.WAIT            ; THESE ARE ALL THE SYSTEM
                                        ; MACRO CALLS
        ..V2..
        .REGDEF
        ;
        ;
PASS1:  ; START PASS1
        MOV     #ISR,@#KWIV             ; THIS IS THE FIRST INTERRUPT SERVICE
                                        ; ROUTINE TO TAKE CARE OF PREFLIGHT
                                        ; DATA
        MOV     #NOINT,@#KWIV+2         ; NO FURTHER INTERRUPT IS ALLOWED
                                        ; WHEN THE INTERRUPT SERVICE ROUTINE
```

```
                                        ; IS EXECUTING (BR7)
        ;
        .FETCH  #HNDR,#DEVICE           ; GET THE DK HANDLER
                                        ; INTO MEMORY
        BCC     1$
        MOV     #FERR,R0                ; SOMETHING'S WRONG
        JMP     FAIL                    ; FATIAL ERROR
1$:     .ENTER  #AREA,#0,#FILE,#FILSIZ  ; CREATE PASS1.TMP ON THE DISK
                                        ; ON I/O CHANNEL 0
                                        ; WITH DEFAULT FILE SIZE
        BCC     2$
        MOV     #EERR,R0                ; CANNOT CREATE NEW FILE
        JMP     FAIL                    ; FATIAL ERROR
2$:     ;       ERASE THE SCOPE AND GET READY
        ;
        MOV     #ERASE,@#VCST           ; ERASE THE 603 SCREEN
        MOV     #VCRDY,@#VCST           ; GET READY, Y MODE, FAST INTENSIFY
        WAIT    VCST                    ; WAIT TILL IT IS READY
        ;
        ; INITILIZE SOME PARAMETERS
        ;
        CLR     X                       ; X-COOR OF THE LPSVC
        CLR     RECNO                   ; BLOCK NUMBER FOR .WRITE MACRO
        CLR     WBUF                    ; FLAG TO IDENTIFY DOUBLE BUFFER
                                        ; 0 MEANS BUFFER 1
                                        ; 1 MEANS BUFFER 2
        ;
        ;
        MOV     #KWENBL,@#KWST          ; START THE CLOCK
13$:    ;
        MOV     #CHAN0,@#ADST           ; SAMPLE ADC CHANNEL 0
        WAIT    ADST                    ; WAIT FOR ADC
        CMP     @#ADBF,#ZEROV           ; SEE IF IT IS A VOLTAGE DROP
        ;       THE ABOVE NEEDS SOME EXPLANATION
        ;       THE PROGRAM WILL EXECUTE IN IDLE
        ;       UNTIL THE PUSH BUTTON AT CHANNEL 0
        ;       FIRES A PULSE
        ;       THAT MEANS THE FIRST WAVEFORM APPEARS
        ;       AND DATA ACQUISITION PHASE WILL START
        BLE     START
        ; OTHERWISE
        ;       ENTER AN IDLE LOOP TO KILL SOME TIME
        ;       THE IDLE TIME DEPENDS ON THE PULSE WIDTH
        ;       OF THE PULSE GENERATOR AT CHANNEL 0
        ;       JUST WANT TO MAKE SURE THAT THE SAME
        ;       PULSE IS NOT READ TWICE
        ;
        MOV     #ICNT1,IN1
3$:     DEC     IN1
        BNE     3$                      ; LOOP
        ;
        ;       NOW SEE IF THE SCREEN IS FULL
        ;       ERASE IT IF NECESSARY
        ;
        CMP     #4095.,X
        BGE     13$                     ; NOT YET
        MOV     #ERASE,@#VCST           ; ERASE SCREEN
        MOV     #VCRDY,@#VCST           ; SET IT UP AGAIN
        WAIT    VCST
        CLR     X                       ; START FROM 0 AGAIN
        BR      13$                     ; LOOP AGAIN
        ;
        ;
        ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THE REAL THING STARTS HERE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
START:  CLR     @#KWST                  ; STOP THE CLOCK FIRST
        MOV     #KWSERV,@#KWIV          ; PUT A NEW INTERRUPT SERVICE
                                        ; ROUTINE TO THE LPSWK'S VECTOR
        MOV     #BUFSIZ,R2              ; R2 IS USED AS A COUNTER FOR BUFFER
        MOV     #BUFF1,R1               ; R1 HAS THE ADDRESS FOR BUFFER1
                                        ; START FILLING BUFFER1 FIRST
        CALL    POLL                    ; GET BCD TIME FOR THE FIRST WAVEFORM
        ; IDLE LOOP TO KILL TIME
        MOV     #ICNT1,IN1
1$:     DEC     IN1
        BNE     1$
        ;
        ;       HERE IS THE MAIN LOOP
        ; THE ROUTINE WILL LOOP FOREVER UNTIL A VOLTAGE FALL IS DETECTED
        ; ON CHANNEL 2, THEN IT WILL CLOSE ALL FILES AND EXIT
        ;
LOOP:   ; THIS IS THE MAIN LOOP
        ;
        MOV     #CHAN0,@#ADST           ; SAMPLE CHANNEL 0
        WAIT    ADST
        CMP     @#ADBF,#ZEROV           ; IS IT A VOLTAGE DROP
        BGT     7$                      ; NO
        CALL    POLL                    ; OTHERWISE, READ IN BCD TIME
7$:     ;
        ;       THIS IS AN IDEL LOOP TO KILL TIME
        ;
        MOV     #ICNT2,IN1
4$:     DEC     IN1
        BNE     4$
        MOV     #CHAN2,@#ADST           ; GET A READING FROM CHANNEL 2
        WAIT    ADST
        CMP     @#ADBF,#ZEROV           ; IS IT A DROP?
        BGT     3$                      ; NO, GO ON
        JMP     EOT                     ; YES, THAT'S ALL FOLKS
3$:     CMP     X,#4095.                ; IS THE SCREEN FULL?
        BLE     LOOP                    ; NO
        MOV     #ERASE,@#VCST           ; YES, ERASE THE SCREEN
        MOV     #VCRDY,@#VCST           ; SET IT UP AGAIN
        WAIT    VCST
        BR      LOOP                    ; GO BACK TO LOOP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       THE MAIN ROUTINE ENDS HERE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       SUBROUTINE TO READ IN TIME CODE
;       NOTE THAT THE LPS USES NEGATIVE LOGIC
;       THEREFORE ALL BIT PATTERNS ARE COMPLEMENTED
;
.SBTTL  POLL TIME
.CSECT  POLL
;
```

```
        ;       TIMING INFORMATION COME IN 4 BYTES OF BCD
        ;       THRU THE DIO PORTS
        ;
POLL:   CLR     @#KWST                  ; STOP THE CLOCK FIRST
        MOV     #EOWF,R4                ; PUT OUT A NULL FIRST
        CALL    STORE                   ; STORE THE END OF WAVEFORM MARK
        MOV     #ERASE,@#VCST           ; ERASE THE SCREEN AS WELL
        CLR     X                       ; RESET THE X-COOR COUNTER
        MOV     #VCRDY,@#VCST           ; SET THE SCOPE UP AGAIN
        ;
        ;       THE DIGITAL I/O USES NEGATIVE LOGIC
        ;       THEREFORE PROGRAM HAS TO SENT OUT THE COMPLEMENTED BIT PATTERN
        ;
        MOV     #177776,@#DROUT         ; A -1 TO PULL IN SECOND
        BIS     #2,@#DRST               ; INPUT THRU DIGITAL INPUT PORT
        MOV     @#DRIN,R4               ; DATA IN R4
        CALL    STORE                   ; STORE IT INTO THE BUFFER
        MOV     #177775,@#DROUT         ; A -2 TO GET MINUTES
        BIS     #2,@#DRST
        MOV     @#DRIN,R4
        CALL    STORE                   ; STORE DATA INTO THE BUFFER
        MOV     #177773,@#DROUT         ; A -4 TO GET HOUR
        BIS     #2,@#DRST
        MOV     @#DRIN,R4
        CALL    STORE
        MOV     #177767,@#DROUT         ; A -10 TO GET DAYS OF THE YEAR
        BIS     #2,@#DRST
        MOV     @#DRIN,R4
        CALL    STORE
        ;
        MOV     #KWENBL,@#KWST          ; RESTART THE CLOCK
        RETURN                          ; GO BACK
        ;
        ;       THE FIRST INTERRUPT SERVICE ROUTINE
        ;       TO HANDLE PRE-FLIGHT DATA
        ;
        .SBTTL  INTERRUPT SERVICE ROUTINES
        .CSECT  ISR
        ; THIS ROUTINE JUST DISPLAY A LINE ON THE SCREEN
        ; WHENEVER A DATA PULSE COME IN
        ; THIS ROUTINE TAKES 128 uS
ISR:    MOV     X,@#VCX
        MOV     #YVAL,@#VCY
        ADD     #XINC,X
        RTI                             ; JUST TO DISPLAY A POINT ON THE SCOPE
        ;
        ;       THIS INTERRUPT SERVICE ROUTINE HANDLES
        ;       IN-FLIGHT DATA
        ;       IT READS THE COUNT FROM LPSKW'S BUFFER/PRESET
        ;       REGISTER AND SAMPLE CHANNEL 1 OF THE ADC
        ;       THEN DISPLAY BOTH OF THEM ON THE SCREEN AND
        ;       THEN STORE THEM INTO THE CURRENT BUFFER IN MEMORY
        ;
KWSERV: MOV     @#KWBP,R4               ; COUNT IN BUFFER/PRESET REGISTER
        CALL    STORE                   ; STORE IT INTO THE CURRENT BUFFER
        MOV     #CHAN1,@#ADST           ; SAMPLE ADC CHANNEL 1
        WAIT    ADST                    ; WAIT TILL IT IS READY
        MOV     @#ADBF,R4               ; GET THE READING
        CALL    STORE                   ; STORE IT INTO BUFFER
        MOV     X,@#VCX                 ; LOAD X-COOR
```

```
        MOV     #YVAL,@#VCY             ; LOAD Y-COOR
        MOV     R4,@#VCY                ; THE ANALOG CHANNEL TOO
        ADD     #XINC,X                 ; INCREMENT X
        RTI                             ; RETURN FROM INTERRUPT
        ;
        ;
        ;       ROUTINE TO HANDLE FATIAL ERRORS
        ;       IT WILL PRINT OUT THE ERROR MESSAGE AND EXIT
        .SBTTL  FAIL
        .CSECT  FAIL
        ;       FATIAL ERROR
        ;
        ;       PRINT ERROR MESSAGE AND EXIT
        ;
        ;
FAIL:   .PRINT                          ; ERROR MESSAGE ADDRESS IN RO
        .EXIT                           ; BYE, BYE, MISS AMERICAN PIE
        ;
        ;
        ;       ENDING ROUTINE
        ;       IT WILL FILL UP THE REST OF THE CURRENT BUFFER
        ;       WITH ASCII "FI" (FOR FINISH), CLOSE ALL FILES
        ;       AND EXIT
        .SBTTL  END OF TRANSMISSION
        .CSECT  EOT
        ;
EOT:    CLR     @#KWST                  ; STOP THE CLOCK
        TST     R2                      ; IS THE CURRENT BUFFER JUST FULL?
        BEQ     1$                      ; IF YES, IT MUST BE A BIG COINCIDENT
        ;       OTHERWISE
        ;       FILL WITH "FI"
2$:     MOV     EOJ,(R1)+               ; MOVE END-OF-JOB MARK
                                        ; WHICH IS "FI"
        DEC     R2
        BNE     2$                      ; UNTIL THE WHOLE BUFFER IS FULL
1$:     TST     WBUF                    ; SEE WHICH BUFFER THIS IS?
        BEQ     3$                      ; 0 MEANS IT IS BUFFER 1
        .WAIT   #0                      ; IN CASE THE LAST WRITE IS NOT FINISH
                                        ; WAIT FOR IT TO COMPLETE
        ; WRITE SYSTEM MACRO
        .WRITW  #AREA,#0,#BUFF2,#BUFSIZ,RECNO
        BCC     5$                      ; WRITE OK?
6$:     MOV     #WERR,RO                ; NO
        JMP     FAIL                    ; FATIAL ERROR
5$:     BR      4$                      ; EXIT
3$:     .WAIT   #0                      ; THIS WRITES OUT BUFFER 1
        .WRITW  #AREA,#0,#BUFF1,#BUFSIZ,RECNO
        BCS     6$                      ; ERROR
4$:     .CLOSE  #0                      ; CLOSE FILE
        .PRINT  #ENDMSG                 ; PRINT ENDING MESSAGE
        .EXIT                           ; EXIT
        ;
        ;
        .SBTTL  STORE DATA
        ;       STORE DATA POINTS USING DOUBLE BUFFER SCHEME
        ;       DATA TO BE STORED IS PASSED IN FROM R4
        ;       R1 IS A POINTER TO THE CURRENT AVALIABLE LOCATION
        ;       R2 CONTAINS THE NUMBER OF FREE LOCATIONS LEFT
        ;       WBUF IS A FLAG TO INDICATE WHICH BUFFER IS  CURRENTLY
        ;       BEING USED  0 = BUFFER 1, 1 = BUFFER 2
        ;       IF NO ROOM IS AVALIABLE IN THE CURRENT BUFFER,
        ;       IT WILL SWITCH BUFFER AND GUARANTEE THE DATA
        ;       POINT MUST BE SAVED
        .CSECT  STORE
```

```
STORE:  TST     R2                      ; ANY ROOM LEFT
        BNE     3$                      ; YES, EVERYTHING IS OK
        ;
        ; OTHERWIES !!!!!
        CLR     @#KWST                  ; ONE BUFFER IS FULL
                                        ; HAS TO WRITE IT OUT AND SWITCH BUFFER
                                        ; *** STOP THE CLOCK ***
        TST     WBUF                    ; SEE WHICH BUFFER IS FULL
        BNE     1$                      ; 1 MEANS BUFFER 2
        .WAIT   #0                      ; IN CASE THE LAST WRITE IS NOT FINISH
        .WRITE  #AREA,#0,#BUFF1,#BUFSIZ,RECNO
        ;       ISSUE AN ASYNCHRONOUS WRITE
        ;       CONTROL WILL PASS BACK TO PROGRAM RIGHT AFTER
        ;       THE WRITE REQUEST IS QUEUED
        ;
        BCS     4$                      ; ERROR
        MOV     #BUFF2,R1               ; RESET THE POINTER TO BUFFER 2
        INC     WBUF                    ; SET FLAG TO INDICATE  BUFFER 2 IS IN USE
        BR      2$
1$:     .WAIT   #0                      ; THIS TIME IS TO WRITE OUT BUFFER 2
        .WRITE  #AREA,#0,#BUFF2,#BUFSIZ,RECNO
        BCS     4$                      ; ERROR
        MOV     #BUFF1,R1               ; RESET POINTER TO BUFFER 1
        CLR     WBUF                    ; INDICATE BUFFER 1 IS IN USE
        BR      2$
4$:     MOV     #WERR,R0
        JMP     FAIL                    ; WRITE ERROR, FATIAL
2$:     ADD     #RECLEN,RECNO           ; UPDATE BLOCK COUNTER
        MOV     #BUFSIZ,R2              ; RESET POINTS COUNTER
        MOV     #KWENBL,@#KWST          ; RESTART THE CLOCK
3$:     MOV     R4,(R1)+                ; STORE POINT INTO BUFFER
        DEC     R2
        RETURN
        ;
        ;
        .SBTTL  DATA DEFINITIONS
        .CSECT  DATA
        ;
        .NLIST
DEVICE: .RAD50/DK /                     ; DEVICE NAME
FILE:   .RAD50/DK PASS1 TMP/            ; FILE NAME
AREA:   .BLKW   10.                     ; AREA FOR WRITE
RECNO:  .WORD   0                       ; WRITE RECORD NUMBER
WBUF:   .BLKW   1                       ; FALG TO INDICATE WHICH BUFFER
EOJ:    .WORD   "FI                     ; END OF JOB MARK
IN1:    .WORD   0                       ; FREE SLOT FOR COUNTING
X:      .WORD   0                       ; X-COOR
        ;
        ;       ERROR MESSAGES
        ;
FERR:   .ASCIZ/ NO DEVICE /             ; FETCH ERROR
EERR:   .ASCIZ/ CANNOT CREATE /         ; ENTER ERROR
WERR:   .ASCIZ/ WRITE ERROR /           ; WRITE ERROR
ENDMSG: .ASCIZ/ END OF PASS1 /          ; ENDING MESSAGE
        .EVEN
BUFF1:  .BLKW   4096.                   ; BUFFER 1, 4K
BUFF2:  .BLKW   4096.                   ; BUFFER 2, 4K
HNDR:   .+2                             ; PLACE TO PUT DK HANDLER
        .LIST
        .END PASS1
```

```
C
C       DUE TO SOME STRANGE RUN TIME SUPPORT OF RT-11 FORTRANIV
C       AN ASSEMBLER MAIN PROGRAM HAVE PROBLEMS TO CALL
C       FORTRAN SUBROUTINES
C       THEREFORE THIS MAIN PROGRAM HAS TO BE USED TO CALL
C       THE ASSEMBLER SUBROUTINE "PASS2" FOR DATA SEGMENTATION
C       AND THE FORTRAN SUBROUTINE "QUEST" FOR HEADER INFORMATION
C
C
C       SET UP COMMON AREA FOR OUTPUT BUFFER
        COMMON /OBUF/ II(4096)
C       LOGICAL UNIT NUMBER 7 IS THE CONSOLE TERMINAL
        WRITE(7,1)
1       FORMAT(10X,'PASS2 OF DATA PROCESSING',/)
C       QUESTION SUBROUTINE
        CALL QUESTN
C       DATA REFINEING AND SEGMENTATION SUBROUTINE
        CALL PASS2
        END
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;                                                    ;
        ;       PASS 2 OF DATA PROCESSING                    ;
        ;                                                    ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;
        ;       THIS PASS READS IN THE DATA OBTAINED FROM
        ;       PASS1 IN 'PASS1.TMP' AND SPLITS IT INTO
        ;       A WAVEFORM BY WAVEFORM BASIS
        ;       IT DECODES THE TIMING INFORMATION FROM
        ;       BCD TO BINARY, INSERTS HEADER INFORMATION
        ;       AND TRIES TO REFINE THE DATA POINTS
        ;       FINALLY IT ROUNDS THE DATA TO THE NEXT 256
        ;       WORD BOUNDARY
        ;
        ;
        ;
        ;       SOME MACRO DEFINITIONS:
                .NLIST
                .MACRO  PUSH    ; SAVE ALL REGISTERS ON STACK
                MOV     %0,-(%6)
                MOV     %1,-(%6)
                MOV     %2,-(%6)
                MOV     %3,-(%6)
                MOV     %4,-(%6)
                MOV     %5,-(%6)
                .ENDM
                .MACRO  POP
                MOV     (%6)+,%5        ; POP ALL REGISTERS
                MOV     (%6)+,%4
                MOV     (%6)+,%3
                MOV     (%6)+,%2
                MOV     (%6)+,%1
                MOV     (%6)+,%0
                .ENDM
                .MACRO  SAVE    A       ; SAVE A ON STACK
                MOV     A,-(%6)
                .ENDM
                .MACRO  RESTOR  A       ; RESTOR A FROM STACK
                MOV     (%6)+,A
                .ENDM
                .MACRO  CALL    A       ; JUMP TO SUBROUTINE A THRU PC
                JSR     %7,A
```

```
        .ENDM
        .MACRO  RETURN              ; RETURN FROM SUBROUTINE
        RTS     %7
        .ENDM
        .LIST
;
; SOME CONSTANT DEFINITIONS
;
;
ERRWD = 52                          ; SYSTEM ERROR WORD
INSIZ = 4096.                       ; INPUT BUFFER SIZE = 4K
OUTSIZ = 4096.                      ; OUTPUT BUFFER SIZE = 4K
F180 = 560.                         ; COUNT CORROSPONDS TO 180 HZ
FEQLIM = 8.                         ; MORE THAN 8 PTS > 180 HZ
SLPLIM = 8.                         ; MORE THAN 8 PTS HAVE NON-RISING SLOPE
; LPS ADDRESSES
ADBF = 170402                       ; ADC DATA REGISTER
INCNT = 16.                         ; INPUT BUFFER BLOCK LENGTH
                                    ; 4K = 16. BLOCKS
HEADLN = 256.                       ; HEADER = 128 WORD LONG
                                    ; = 256. BYTES
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                ;
;                                                ;
;       MAIN PROGRAM STARTS HERE                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CANNOT USE THE NAME "PASS2" BECAUSE THE CALLING FORTRAN
; PROGRAM REQUIRES THIS NAME
; OTHERWISE WILL GET MULTIPLE DEFINITION ERROR FROM LINK
.TITLE PASS20
.SBTTL PASS2 OF DATA PROCESSING
;       THIS PASS READS IN DATA  OBTAINED FROM PASS1
;       INTO THE INPUT BUFFER, DECODES 4 WORDS(ONLY THE LOWER BYTE
;       OF EACH WORD) OF TIMING INFORMATION INTO BINARY FORM
;       SPLIT THE INCOMING DATA INTO INDIVIDUAL WAVEFORMS
;       AND THROWS OUT UNNECESSARY DATA ( DATA THAT HAS A MAJORITY
;       OF NON-RISING SLOPES AND A MAJORITY OF FREQUENCIES > 180 HZ)
;       FINALLY IT ROUNDS THE DATA INTO THE NEXT 256 WORD BOUNDARY
;       AND WRITE THE WAVEFORM TO THE OUTPUT FILE "PASS2.TMP"
;
;
; SYSTEM MACRO CALLS
;
;
.MCALL  ..V2..,.REGDEF
.MCALL  .FETCH,.ENTER,.LOOKUP
.MCALL  .READW,.WRITW
.MCALL  .EXIT,.CLOSE,.PRINT
; ENRTY POINT
.GLOBL PASS2
;
;
..V2..
.REGDEF
;
;
; CONTROL SECTION DEFINITION
.CSECT  PASS20
; GET RK05 HANDLER IN MEMORY
```

```
PASS2:  .FETCH   #HNDR,#DEVICE
        BCC      1$                      ; FETCH OK?
        MOV      #FERR,R0                ; NO, FATIAL ERROR
        JMP      FAIL                    ; FETCH FAIL
1$:     .LOOKUP  #INAREA,#0,#INFIL       ; OPEN 'PASS1.TMP' ON CHANNEL 0
                                         ; FOR INPUT
        BCC      2$                      ; IS IT OK?
        TSTB     @#ERRWD                 ; WHAT'S WRONG?
        BEQ      3$                      ; 0 MEANS CHANNEL ACTIVE
        MOV      #NERR,R0                ; NO FILE
        JMP      FAIL
3$:     MOV      #AERR,R0                ; CHANNEL IS ACTIVE
        JMP      FAIL
2$:     .ENTER   #OAREA,#1,#OUTFIL,#-1   ; CREATE OUTPUT FILE
                                         ; "PASS2.TMP" ON DISK
        BCC      4$                      ; ENTER OK?
        MOV      #EERR,R0                ; CREATE ERROR
        JMP      FAIL
        ;
        ;
4$:
        ;        INITILIZE SOME PARAMETERS
        ;
        ;
        .SBTTL   INIT PARAM              ; INIT PARAMETERS
        ;
        CLR      INBLK                   ; INPUT BLOCK COUNTER
        CLR      OUTBLK                  ; OUTPUT BLOCK COUNTER
        MOV      #INBUF,R1               ; R1 HAS THE ADDRESS OF THE
                                         ; INPUT BUFFER
        MOV      #INSIZ,R2               ; R2 HAS THE SIZE OF THE INPUT BUFFER
        ; BUT THE OUTPUT BUFFER IS ACTUALLY 4K - 128 WORDS (HEADER)
        ; PUT THIS NUMBER ON "OUTCNT"
        MOV      #OUTSIZ - 128.,OUTCNT   ; 4K - 128. WORDS
        CLR      ENDFLG                  ; END FILE FLAG
        ;
        ;
        .SBTTL   PROCESS HEADER
        ;
        ; THIS IS DONE BY THE FORTRAN SUBROUTINE "QUEST"
        ;
        ;
        ;
        MOV      #OUTBUF,R3              ; R3 POINTS TO BEGINNING OF OUTBUF
        ; BEGIN AT WHERE A ZERO IS INSERTED
        ;
        ; SKIP THE HEADER UNTIL THE POSITION IN WHICH THE "0"
        ; IS PLACED
        ADD      #236.,R3
        CLR      (R3)+                   ; PUT A ZERO IN THERE
        MOV      R3,BADDR                ;SAVE THIS ADDRESS FOR LATER USE
        ;
        .SBTTL   READ IN THE 1'ST BUFFER
        ; READ IN THE FIRST BUFFER JUST TO GET STARTED
        .READW   #INAREA,#0,R1,R2,INBLK
        BCC      6$                      ; READ ERROR ?
        MOV      #RERR,R0
        JMP      FAIL                    ; FATIAL
6$:     ADD      #INCNT,INBLK            ; UPDATE INBLK
        ;
        ; THE REAL PROCESSING STARTS HERE
        .SBTTL START                     ; START PROCESSING
        ;
        ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
        .CSECT START
LOOP:   CALL    READ                    ; READ IN THE 1'ST POINT
                                        ; RETURNS IN R4
NOREAD: TST     R4                      ; UNTIL IT FINDS A ZERO
                                        ; WHICH IS THE END-OF-WAVEFORM MARK
        BEQ     1$
        BR      LOOP                    ; OTHERWISE JUST KEEP LOOPING
1$:     MOV     R4,(R3)+                ; PUT IT IN THE OUTPUT BUFFER FIRST
        CLR     ZFLAG                   ; CLEAR THIS FLAG
        CALL    READ                    ; THE 1'ST TIMING WORD RETURNS IN R4
                                        ; ONLY THE LOWER BYTE IS USED
        CALL    DECODE                  ; DECODE IT INTO BINARY FORM
        MOV     R4,SEC                  ; SAVE IT
        CALL    READ                    ; NEXT
        CALL    DECODE
        TST     R4                      ; IS IT ZERO?
        BEQ     2$
3$:     ADD     #60.,SEC                ; CONVERT MINUTES TO SECONDS
        DEC     R4
        BNE     3$
        ; SAVE IT IN "SEC"
2$:     CALL    READ                    ; NEXT
        COMB    R4                      ; SINCE LPS IS IN NEGATIVE LOGIC
                                        ; THEREFORE COMPLENT IT
        RORB    R4                      ; IS IT SET?
        BCC     4$
        ADD     #3600.,SEC              ; YES, ADD 3600 SECONDS
4$:     MOV     SEC,R4                  ; NOW "SEC" HAS THE BINARY SECONDS
                                        ; OF THE BCD TIME CODE
        MOV     R4,(R3)                 ; PUT IT INTO THE OUTPUT BUFFER
        ADD     #16.,R3                 ; SKIP TO THE BEGINNING POSITION
                                        ; FOR HOLDING DATA POINTS
        ; LED ROUTINE WILL DISPLAY THE CONTENT OF R4 ON THE LED DISPLAY
        ; ON THE LPS IN DECIMAL FORM
        CALL    LED
        ; READ IN THE DAYS OF YEAR, THIS IS NOT USED AT PRESENT
        CALL    READ
        ;
        ;
        ;
        ; THE FOLLOWING SECTION OF CODE IS FOR REFINING THE DATA POINTS
        ; IT READS IN 16. PAIRS OF POINTS AT A TIME
        ; IF THIS 16. PAIRS OF POINTS SATISFY:
        ; A) HAVE A MAJORITY OF POINTS > 180 HZ AND
        ; B) HAVE A MAJORITY OF NON-RISING SLOPES
        ; THEN THE REST OF THE DATA POINTS IN THE WAVEFORM IS
        ; VERY LIKELY TO BE USELESS FOR CALCULATION BECAUSE
        ; CONDUCTIVITY CALCULATIONS INVOLVES ONLY THE RISING
        ; SLOPES OF THE WAVEFORMS
        ; THEREFORE THE REST OF THE DATA POINTS WILL BE DISCARDED
        ; TO SAVE STORAGE SPACE
        ;
READ10: CLR     FEQFLG                  ; THIS FLAG IS USED TO INDICATE HOW MANY
                                        ; POINTS ARE ABOVE 180 HZ
        CLR     SLPFLG                  ; THIS FLAG IS USED TO INDICATE HOW MANY
                                        ; SLOPES ARE NON-RISING
        MOV     #31.,TSTCNT             ; 31 POINTS LEFT
        CALL    READ                    ; 30 LEFT (15 PAIRS)
        TST     R4
        ; IF IT EQUALS TO 0 MEANS THATS THE END OF THE WAVEFORM
        ; NOTHING HAS BEEN THROWN OUT
```

```
        BEQ     SAVE10                  ; TAKE CARE OF THE ROUNDING
        MOV     R4,(R3)+                ; SAVE IT IN THE OUTPUT BUFFER
        DEC     OUTCNT                  ; UPDATE THE COUNTER
        ; HERE IS A LITTLE TRICK TO DETERMINE IF THE SLOPE IS RISING
        ; OR FALLING
        ; SINCE THE COUNT IS INVERSELY PROPORTIONAL
        ; TAKE THE DIFFERENCE BETWEEN 2 CONSECUTIVE COUNT
        ; DEPENDING ON WETHER THE DIFFERENCE IS POSITIVE OR NEGATIVE
        ; CAN DETERMINE THE SLOPE IS RISING OR FALLING
        MOV     R4,R5
                                        ; HAS TO GET THE 1'ST POINT IN
        CMP     #F180,R4                ; SEE IF IT IS > 180 HZ
        BLT     5$
        INC     FEQFLG                  ; IF > INCREMENT FLAG
        ; NOW ENTER THE LOOP FOR THE REST OF THE 31 POINTS
5$:     CALL    READ                    ; READ IN R4
        MOV     R4,(R3)+                ; SAVE IN OUTBUF
        DEC     OUTCNT
        DEC     TSTCNT                  ; UPDATE BOTH COUNTERS
        CALL    READ
        TST     R4
        BEQ     SAVE10
        MOV     R4,(R3)                 ; ANALOG TOO
        DEC     OUTCNT
        ; COMPARE FREQUENCY AND SLOPE
        CMP     #F180,R4
        BLT     6$
        INC     FEQFLG
6$:     SUB     R5,R4           ; IF LAST COUNT > PRESENT COUNT
                                ; => RISING SLOPE
        BLT     7$
        INC     SLPFLG
7$:     MOV     (R3)+,R5
        DEC     TSTCNT
        BNE     5$
        ; SEE IF BOTH LIMITS EXCEED
        ; IF THEY DO, THE REST OF THE POINTS CAN BE THROWN OUT
        CMP     #FEQLIM,FEQFLG
        BGE     READ10
        CMP     #SLPLIM,SLPFLG
        BGE     READ10
        ; IF EXECUTION REACHES HERE, BOTH LIMITS HAVE BEEN EXCEEDED
        ; THE ZERO FLAG IS USED TO INDICATE WHETHER IT NEEDS A READ OR NOT
        ; IF THE "0" HAS ALREADY BEEN READ, THERE IS NO NEED TO READ AGAIN
        ; THIS IS TO KEEP THE PROPER ALIGNMENT OF DATA
        INC     ZFLAG           ; BACK SPACE 0 OR NOT
        ; THE FOLLOWING SECTION OF CODE WRAPS UP THE PIECES
        ; IT ROUNDS UP THE REST OF THE BUFFER TO THE NEXT
        ; 256 WORDS BOUNDARY AND WRITE IT TO THE DISK
SAVE10:
        MOV     #OUTSIZ,R5              ; DETERMINE HOW MANY POINTS HAVE
                                        ; BEEN ENTERED
        SUB     OUTCNT,R5
        MOV     R5,R4
        SUB     #128.,R4
        ; DIVIDED BY 2 GIVES NUMBER OF POINT PAIRS
        ASR     R4                      ; THIS IS THE NUMBER OF PAIRS OF POINTS
        MOV     R4,NPTS                 ; SAVE IT
        CLR     R4
1$:     INC     R4                      ; CALCULATE TO THE NEXT 256 WORDS
        SUB     #256.,R5
        BGT     1$
        TST     R5
        BEQ     2$                      ; OR MAY BE IT IS JUST RIGHT AT
```

```
                                        ; 256 WORDS BOUNDARY ??
3$:     CLR     (R3)+                   ; CLEAR THE REST
        INC     R5
        BNE     3$
2$:     MOV     R4,NBLK                 ; SAVE THE NUMBER OF BLOCKS
        CLR     R5
        ; THE SYSTEM MACRO WRITE REQUIRES THE BUFFER SIZE IN WORDS
        ; THEREFORE HAS TO CALCULATE 256 * NUMBER OF BLOCKS
4$:     ADD     #256.,R5                ; GET BACK NBLKS*256 TO WRITE
        DEC     R4
        BNE     4$
        ; WRITE (NUMBER OF BLOCKS) AND (NUMBER OF DATA POINT PAIRS)
        ; TO THE HEADER
        MOV     #OUTBUF,R4
        ADD     #248.,R4
        MOV     NPTS,(R4)
        ADD     #4,R4
        MOV     NBLK,(R4)
        ; WRITE OUT THE WAVEFORM TO DISK
        .WRITW  #OAREA,#1,#OUTBUF,R5,OUTBLK
        ADD     NBLK,OUTBLK             ; UPDATE COUNTER
        MOV     #OUTSIZ-128.,OUTCNT     ; RESET OUTPUT SIZE COUNT
        MOV     BADDR,R3                ; RESET BEGINNING ADDRESS
        TST     ZFLAG                   ; SEE IF I NEED TO BACK SPACE 0
        BEQ     5$
        JMP     LOOP
5$:     CLR     R4
        JMP     NOREAD
        ;
        ; END
        ;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ; END PASS2 MAIN ROUTINE
        ;
        ;
        ; THIS ROUTINE WILL RETURN A DATA POINT IN R4
        ; IT AUTOMATICALLY MAINTAINS INPUT BUFFERING
        ; A 4K INPUT BUFFER IS USED
        ; IF DATA FROM INPUT BUFFER IS EXAUSTED
        ; IT WILL READ IN ANOTHER BUFFER FROM THE FILE "PASS1.TMP"
        ; UNTIL END OF FILE IS REACHED
        .SBTTL  READ
        ;
        ;
        .CSECT  READ
READ:   TST     R2                      ; R2 HAS INPUT BUFFER COUNT
        ; IS THERE ANYTHING LEFT ?
        BNE     1$                      ; YES
        TST     ENDFLG                  ; IS THIS THE LAST BUFFER OF THE FILE
        BEQ     2$
        JMP     EOF                     ; YES, LAST BUFFER
        ; OTHERWISE, READ IN ANOTHER BUFFER FROM "PASS1.TMP"
2$:     MOV     #INBUF,R1
        .READW  #INAREA,#0,R1,#INSIZ,INBLK
        BCC     3$                      ; READ OK?
        TSTB    @#ERRWD                 ; WHAT'S WRONG?
        BEQ     4$
        MOV     #RERR,R0                ; READ ERROR
        JMP     FAIL                    ; READ ERROR, FATIAL
4$:     TST     R0
        BNE     5$
        JMP     EOF
5$:
```

```
        MOV     R0,R2                   ; END OF FILE, R0 CONTAINS ACTUAL
                                        ; NO OF POINTS READ
        INC     ENDFLG                  ; SET END-OF-FILE FLAG
        BR      1$                      ; GO ON WITH THE LAST BUFFER
3$:     ADD     #INCNT,INBLK            ; UPDATE INPUT BLOCK COUNTER
                                        ; 4K = 16 BLOCKS
        MOV     #INSIZ,R2               ; RESET COUNTER
1$:     MOV     (R1)+,R4                ; RETURN DATA IN R4
        DEC     R2                      ; UPDATE COUNTER
        CMP     R4,EOJ                  ; IS IT THE END OF FLIGHT MARK
        BNE     6$
        JMP     EOF                     ; YES, THE END
6$:
        RETURN
        ;
        ;
        ;
        ; DECODE BCD TIME
        ;
        .SBTTL DECODE
        ; DECODE THE LOWER BYTE OF R4 WHICH CONTAINS 2
        ; BCD DIGITS AND RETURNS THE BINARY RESULT IS R4
        ; THE BCD NUMBER IS IN NEGATIVE LOGIC
        ; THEREFORE HAS TO BE COMPLEMENTED FIRST
        .CSECT  DECODE
        ;
        ; THIS MIGHT LOOK CLUMSY BUT IT IS THE MOST EFFICENT WAY
        ;
        ; BELIEVE IT OR NOT
        ;
DECODE: CLR     R5                      ; R5 IS TO HOLD THE RESULT TEMPORARY
        COM     R4
        RORB    R4                      ; R4 HAS THE BCD NUMBER
        BCC     1$
        INC     R5                      ; R5 WILL HAVE THE BINARY NUMBER
1$:     RORB    R4
        BCC     2$
        ADD     #2,R5
2$:     RORB    R4
        BCC     3$
        ADD     #4,R5
3$:     RORB    R4
        BCC     4$
        ADD     #8.,R5
4$:     RORB    R4
        BCC     5$
        ADD     #10.,R5
5$:     RORB    R4
        BCC     6$
        ADD     #20.,R5
6$:     RORB    R4
        BCC     7$
        ADD     #40.,R5
7$:     RORB    R4
        BCC     8$
        ADD     #80.,R5
8$:     MOV     R5,R4                   ;RETURN RESULT IN R4
        RETURN
        ;
        ;
        ; CLOSE ALL FILES, PRINT OUT ENDING MESSAGE AND EXIT
        .SBTTL  EOF                     ; END-OF-FILE
EOF:
        .CLOSE  #0                      ; CLOSE INPUT FILE
```

```
        .CLOSE  #1                      ; CLOSE OUTPUT FILE
        .PRINT  #ENDMSG                 ; END MESSAGE
        .EXIT
;
;
        .SBTTL  FATIAL ERROR
        ; PRINT OUT ERROR MESSAGE AND EXIT
;
FAIL:   .PRINT                          ; MESSAGE ADDRESS IN RO
        .EXIT
;
        .SBTTL LED
        .CSECT LED
        ; R4 CONTAINS THE BINARY NUMBER TO BE DISPLAYED
        ; DIVIDE IT INTO POSITIONAL FORM(DECIMAL) AND PUT IT OUT ON THE LED
        ; DISPLAY ON THE LPS
LED:    CLR     LED1
        ; CLEAR ALL 6 DIGITS FIRST
        MOV     #17,@#ADBF
        MOV     #417,@#ADBF
        MOV     #1017,@#ADBF
        MOV     #1417,@#ADBF
        MOV     #2017,@#ADBF
        MOV     #2417,@#ADBF
1$:     MOV     #-1,LED2
2$:     INC     LED2
        SUB     #10.,R4                 ; GET REMAINDER
        BGE     2$
        ADD     #10.,R4
        ; SET UP A WHOLE WORD AND GO
        MOVB    R4,LED3
        MOVB    LED1,LED3+1
        MOV     LED3,@#ADBF             ; ILLUMINATE
        INC     LED1
        MOV     LED2,R4
        TST     R4
        BNE     1$
        RETURN
        ; END LED
LED1:   .WORD 0
LED2:   .WORD 0
LED3:   .WORD 0
;
        .SBTTL  DATA DEFINATIONS
        .CSECT  DATA
EOJ:    .WORD   "FI                     ; FINISH MARK
INBLK:  .WORD   0                       ; INPUT BLOCK COUNT
OUTBLK: .WORD   0                       ; OUTPUT BLOCK COUNT
OUTCNT: .WORD   0                       ; OUTPUT BUFFER WORD COUNT
TSTCNT: .WORD 0                         ; TEST COUNT FOR 31 POINTS
BADDR:  .WORD 0                         ; TO HOLD BEGIN ADDRESS FOR OUTBUF
SEC:    .WORD   0                       ; STORE SECOND
NBLK:   .WORD 0                         ; NUMBER OF BLOCKS
NPTS:   .WORD 0                         ; NUMBER OF POINTS
ZFLAG:  .WORD   0                       ; FREQ/SLOPE FLAG
ENDFLG: .WORD   0                       ; END-OF-FILE FLAG
SLPFLG: .WORD   0                       ; SLOPE FLAG
FEQFLG: .WORD   0                       ; FREQUENCY FLAG
INAREA: .BLKW   10.                     ; SYSTEM MACRO AREA
OAREA:  .BLKW   10.                     ; FOR OUTPUT
INBUF:  .BLKW   4096.                   ; INPUT BUFFER 4K
;
        ; NAME THE OUTPUT BUFFER BY CSECT NAME OBUF
        ; TO MAKE IT KNOWN TO FORTRAN SUBROUTINE QUESTN
```

```
        ;
        .CSECT OBUF
OUTBUF: .BLKW   4096.                   ; OUTPUT BUFFER 4K
        .NLIST
        .CSECT DATA
INFIL:  .RAD50/DK PASS1 TMP/            ; INPUT FILE NAME
OUTFIL: .RAD50/DK PASS2 TMP/            ; OUTPUT FILE NAME
DEVICE: .RAD50/DK /
FERR:   .ASCIZ/NO DEV/                  ; FETCH ERROR
NERR:   .ASCIZ/NO FILE/                 ; NO FILE
AERR:   .ASCIZ/CHAN ACT/                ; CHANNEL ACTIVE
EERR:   .ASCIZ/CANT CREATE/             ; ENTER ERROR
RERR:   .ASCIZ/READ ERR/                ; READ ERROR
OERR:   .ASCIZ/NO ROOM/                 ; OUTPUT BUFFER FULL
WERR:   .ASCIZ/WRITE ERR/               ; WRITE ERROR
ENDMSG: .ASCIZ/END PASS2/               ; END MESSAGE
        .EVEN
HNDR:   .+2
        .LIST
        .END PASS2
      SUBROUTINE QUESTN
C       DECLARE COMMON AREA
      COMMON /OBUF/DATE(2),STYPE,SNUMB,LASITE,RF,RCAL,R1,R2,L,DFDTCL,
     1DTSW,DVSW,VSWN,VSWP,DUM1,IDEN,TVO,ALT,VERVEL,ISAT,DUM2,
     2SIG(8,4),DUM3,DUM4,DUM5,ZERO,TIME(2),NPTS,NBLKS
        REAL LASITE,L,ISAT
        INTEGER*2 IDEN(8)
      INTEGER*2 STYPE,SNUMB
        INTEGER*4 NPTS,NBLKS
C
C  SUBROUTINE TO FILL INFORMATION BLOCK FOR HEADER OF EACH  RECORD
C
        IN = 5
        IOUT = 7
        ALT = -1.0
        DO 100 I = 1,8
        DO 100 J = 1,4
100     SIG(I,J) = 0.0
      WRITE(IOUT, 1)
    1 FORMAT(' PROGRAM TO INPUT HEADER INFORMATION FOR EACH FLIGHT')
C
C  GET DATE
C
      WRITE(IOUT, 2)
    2 FORMAT(' DATE--DDMMMYY')
      READ(IN,3) DATE(1),DATE(2)
    3 FORMAT(2A4)
C
C  GET SENSOR TYPE
C
      WRITE(IOUT, 4)
    4 FORMAT(' SENSOR TYPE BP FOR BLUNT PROBE-- GC FOR GERDIEN')
      READ(IN,5)  STYPE
    5 FORMAT(A2)
C
C  GET SENSOR NUMBER
C
      WRITE(IOUT, 6)
    6 FORMAT(' SENSOR NUMBER-- I3')
      READ(IN,7) SNUMB
    7 FORMAT(I3)
C
C  GET LAUNCH SITE
C
```

```
      WRITE(IOUT,8)
    8 FORMAT(' LAUNCH SITE -- XXXX')
      READ(IN,9) LASITE
    9 FORMAT(A4)
C
C  GET FEEDBACK RESISTOR
C
      WRITE(IOUT,10)
   10 FORMAT(' FEEDBACK RESISTOR SIZE -RF- F7.2')
      READ(IN,11) RF
   11 FORMAT(E7.2)
C
C  GET CALIBRATION RESISTOR
C
      WRITE(IOUT,12)
   12 FORMAT(' CALIBRATION RESISTOR SIZE -RCAL-F7.2')
      READ(IN,13) RCAL
   13 FORMAT(E7.2)
C
C  COLLECTOR RADIUS
C
      WRITE(IOUT,14)
   14 FORMAT(' COLLECTOR RADIUS-R FOR BP--RI FOR GC  F4.1')
      READ(IN,15) R1
   15 FORMAT(F4.1)
C
C  GUARD RADIUS
C
      WRITE(IOUT,16)
   16 FORMAT(' GUARD OR OUTER PLATE RADIUS- R FOR BP-RO FOR GC  F4.1')
      READ(IN,17) R2
   17 FORMAT(F4.1)
C
C  ELECTRODE LENGTH
C
      WRITE(IOUT,18)
   18 FORMAT(' ELECTRODE LENGTH IN CM - ENTER 0 FOR BP  F4.1')
      READ(IN,19) L
   19 FORMAT(F4.1)
C
C  DF/DT CAL
C
      WRITE(IOUT,20)
   20 FORMAT(' DF/DT CAL  F6.2')
      READ(IN,21) DFDTCL
   21 FORMAT(F6.2)
C
C  DELTA TIME SWEEP
C
      WRITE(IOUT,22)
   22 FORMAT(' DELTA TIME SWEEP  F6.2')
      READ(IN,23) DTSW
   23 FORMAT(F6.2)
C
C  DELTA VOLTAGE SWEEP
C
      WRITE(IOUT,24)
   24 FORMAT(' DELTA VOLTAGE SWEEP  F6.2')
       READ(IN,25) DVSW
   25 FORMAT(F6.2)
C
C  NEGATIVE MAXIMUM VALUE OF VOLTAGE SWEEP
C
      WRITE(IOUT,26)
```

```
   26 FORMAT(' NEGATIVE MAXIMUM VALUE OF SWEEP VOLTAGE  F6.2')
      READ(IN,27) VSWN
   27 FORMAT(F6.2)
C
C  POSITIVE MAXIMUM VALUE OF VOLTAGE SWEEP
C
      WRITE(IOUT,28)
   28 FORMAT(' POSITIVE MAXIMUM VALUE OF SWEEP VOLTAGE  F6.2')
      READ(IN,29) VSWP
   29 FORMAT(F6.2)
C
C  SPECIAL IDENTIFICATION OR INFORMATION
      WRITE(IOUT,30)
   30 FORMAT(' SPECIAL IDENTIFICATION OR INFORMATION  8A2')
      READ(IN,31) (IDEN(I9),I9=1,8)
   31 FORMAT(8A2)
C
C  END OF QUESTION AND ANSWER SUBROUTINE
      WRITE(IOUT,32)
   32 FORMAT(' END OF QUESTION AND ANSWER SECTION OF PROGRAM')
      RETURN
      END
C
C        MAIN PROGRAM FOR PASS3 OF DATA PROCESSING
C
C        PROGRAM LAY OUT:
C                THIS PROGRAM WILL CALL THE FOLLOWING SUBROUTINES
C
C        SUBROUTINE INIT - INITILIZE ALL THE PROGRAM PARAMETERS INCLUDING
C                I/O LOGICAL UNIT NUMBERS AND CONSTANTS, OPEN INPUT
C                AND OUTPUT FILES.
C        SUBROUTINE LED - DISPLAY TIMING INFORMATION ON THE LED DISPLAY
C                OF THE LPS UNIT.
C        SUBROUTINE READIN - READ IN THE NEXT WAVEFORM. SINCE WAVEFORMS
C                ARE IN VARIABLE LENGTH, THEREFORE IT HAS TO READ IN THE 1'ST
C                256 WORD, FIND OUT THE LENGTH OF THE WAVEFORM FROM THE HEADER
C
C        SUBROUTINE DISPR - DISPLAY DATA ON THE TEKTRONIX 603 SCOPE
C                ACCORDING TO THE FREQUENCY LIMITS. A FLAG IS USED TO
C                INDICATE WHEATHER THE ANALOG CHANNEL IS TO BE DISPLAIED OR NOT.
C
C        SUBROUTINE EXPAND - DETERMINE WHEATHER THE WAVEFORM IS TO BE EXPANDED
C                FOR DIFFERENT LIMITS.
C
C        SUBROUTINE SLOPE - GET THE BREAK POINTS OF THE WAVEFORM FROM THE
C                TERMINAL AND PERFORM A WEIGHTED LEAST SQUARE STRAIGHT LINE
C                FIT TO THE SLOPE OF THE WAVEFORM, THEN DISPLAYS THE FITTED
C                STRAIGHT LINE ON THE SCREEN
C
C        SUBROUTINE ENDW - END OF WAVEFORM ROUTINE. CALCULATE THE CONDUCTIVITY
C                VALUE FROM THE SLOPE, RESET THE PARAMETERS, WRITE THE HEADER
C                BACK TO THE INPUT FILE AND WRITE CONDUCTIVITY INFO TO OUTPUT FILE
C
C
C
C        DATA BASE DEFINITIONS
C
C
C        COMMON DATA -  INTEGER ARRAY IDATA(4096) WHICH
C                IS THE INPUT BUFFER TO HOLD ALL THE DATA POINTS
C        COMMON PARM - CONTAIN ALL THE PARAMETERS AND CONSTANTS
C                IIN - LOGICAL UNIT NUMBER FOR INPUT FILE (19)
C                IOUT - LOGICAL UNIT NUMBER FOR OUTPUT FILE (20)
C                INEXT - RECORD NUMBER IN INPUT FILE TO HOLD NEXT RECORD NO.
```

```
C               ILAST - RECORD NUMBER FOR THE BEGINNING OF THE LAST WAVEFORM
C               ISIZ - MAXIMUM SIZE FOR THE INPUT FILE ( REQUIRED BY THE
C                   RT-11 DEFINE FILE STATEMENT )
C               A - SLOPE FOR DEVICE COORDINATE CONVERSION
C               B - INTERCEPT FOR DEVICE COORDINATE CONVERSION
C               C - CONSTANT FOR CALCULATING CONDUCTIVITIES
C               D - RATIO OF TIME TO DEVICE COORDINATES
C
C
C
C
C*******************************************************
C                                                      *
C       PASS3 OF DATA PROCESSING                       *
C                                                      *
C                                                      *
C*******************************************************
C
C       THE MAIN PROGRAM WILL SET LOGICAL UNIT NUMBER 5 AS THE STANDARD
C       INPUT ( TERMINAL ) AND 6 AS THE STANDARD OUTPUT ( TERMINAL )
C
C       COMMON DECLEARATION
C
        COMMON /DATA/ IDATA(4096)       ! HOLD INPUT DATA
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D ! PARAMETERS
C       FOR ALTITUDE INTERPOLATION
        COMMON /ALTITU/ NALT,T(100),Z(100)
C       EQUIVALENCE PART OF THE HEADER INFORMATION
C       TIMING, NUMBER OF (PAIR) POINTS, NUMBER OF 256 WORD BLOCKS
        EQUIVALENCE (ITIME,IDATA(121)),(NPTS,IDATA(125)),(NBLK,IDATA(127))
        DATA ICHAR,JCHAR /'  ','Y '/    ! NOTE UNIX WILL PUT 'Y' IN THE LOWER BYTE
C
C       START
C
C       5 FOR TERMINAL INPUT IS DEFAULT FOR RT-11
        CALL ASSIGN(6,'TT:/N')          ! ASSIGN 6 FOR TERMINAL OUTPUT
C       UNIX DO NOT REQUIRE THIS, THIS IS DEFAULT
C
        NWAVE = 1                       ! WAVEFORM NUMBER
        WRITE(6,1)                      ! WRITE OUT MESSAGE
1       FORMAT('**** PASS3 OF DATA PROCESSING ****')
C
C       INITILIZE PARAMETERS
        CALL INIT
C       DETERMINE WHICH WAVEFORM TO START
        WRITE(6,2)
2       FORMAT('WHICH WAVEFORM TO START?,I4 FORMAT')
        READ(5,3) NSTART
3       FORMAT(I4)
        IF( NSTART .LE. 1 ) GOTO 10     ! A <CR> IS INTERPERATE AS 0
C       OTHERWISE SKIP WAVEFORMS
        DO 100 I = 1,NSTART-1           ! SKIP NSTART - 1 WAVEFORMS
C       READIN WILL READ IN THE NEXT WAVEFORM
100     CALL READIN
        NWAVE = NSTART                  ! UPDATE WAVEFORM NUMBER
C
C       THE MAIN LOOP IS HERE
C
10      CALL READIN                     ! READ IN THE DESIRED WAVEFORM
C       THIS IS WHY NSTART-1 WAVEFORM IS SKIPPED BECAUSE THIS CALL TO
C       READIN WILL GET TO THE RIGHT WAVEFORM
        CALL LED(ITIME)                 ! DISPLAY THE TIMING OF THIS WAVEFORM
C       WRITE SOME INFORMATION OUT
        WRITE(6,4) NWAVE,ITIME,NPTS,NBLK
```

```
4       FORMAT('WAVEFORM',I4,5X,'TIME =',I5,5X,
     1 'DATA POINTS =',I5,5X,'BLOCKS =',I4)
C       IF THIS IS AN EMPTY WAVEFORM, SKIP TO THE NEXT ONE
        IF( NPTS .LE. 10) GOTO 11        ! UPDATE COUNTER AND READ AGAIN
C       DISPLAY WAVEFORM WITH/WITHOUT ANALOG CHANNEL
        CALL DISPR(1,NPTS,0,200,1)       ! ALL POINTS, 0-200 HZ, FLAG = 1
        CALL SLOPE                       ! GET BREAK POINTS AND FIT SLOPE
C       SEE IF THERE ARE ANY MORE DATA FROM THE INPUT FILE
        IF( INEXT .GE. ISIZ ) GOTO 1000 ! END
C       OTHERWISE !
11      NWAVE = NWAVE + 1                ! UPDATE WAVEFORM NUMBER
        WRITE(6,5)
5       FORMAT('CONTINUE ?')
        READ(6,6) ICHAR                  ! GET ANSWER FROM THE TERMINAL
6       FORMAT(A1)
        IF( ICHAR .EQ. JCHAR ) GOTO 10   ! YES, CONTINUE
C       OTHERWISE
1000    WRITE(6,7)
7       FORMAT('**** END OF PASS3 ****')        ! END MESSAGE
        END
C
C       THIS PROGRAM MODULE CONTAINS MOST OF THE SUBROUTINES
C       PASS3 CALLS
C
C
C
C
C
C       SUBROUTINE INIT INITILIZES PARAMETERS AND OPEN INPUT AND
C       OUTPUT FILES
C
        SUBROUTINE INIT
C       COMMON DEFINITIONS
        COMMON /DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
        COMMON /ALTITU/ NALT,T(100),Z(100)
C       INITILIZE I/O LOGICAL UNIT NUMBERS
        IIN = 19
        IOUT = 20
C       INPUT ALTITUDE INFO
        WRITE(6,5)
5       FORMAT('INPUT TIME-ALTITUDE FILE NAME',/)
C       THIS IS A RT-11 SYSTEM SUBROUTINE
        CALL ASSIGN(21,'???.???',-1,'RDO')
        DO 100 I = 1,100
C       READ IN ASCII RADAR DATA FILE
        READ(21,10,END=101) T(I),Z(I)
100     CONTINUE
10      FORMAT(2F8.2)
C       OPEN I/O FILES
C
C       GET THE NUMBER OF RADAR DATA POINTS
101     NALT = I-1
C       UNIX WILL CALL "SETFIL" WHICH DO THE SAME THING
C
C
C       OPEN "PASS2.TMP"
        CALL ASSIGN(IIN,'PASS2.TMP',9,'OLD')     ! INPUT FILE
C       CREATE "PASS3.TMP", ASCII OUTPUT
        CALL ASSIGN(IOUT,'PASS3.TMP',9,'NEW')    ! CREATE OUTPUT FILE
C       A AND B ARE CONSTANTS TO CONVERT FREQUENCY TO DEVICE COORDINATE
C       ON THE TEKTRONIX 603
        A = 2.0475000E1
        B = 0.0          ! START WITH 0 - 200 HZ
```

```
        WRITE(6,1)
1       FORMAT(' NUMBER OF BLOCKS IN PASS2.TMP, I4 FORMAT')
        READ(5,2) N
2       FORMAT(I4)
C       RT-11 DEFINE FILE STATEMENT REQUIRES THIS
        ISIZ = N                   ! SAVE FILE SIZE
C       THIS THE IS RT-11 FORMAT
C       SOME OTHER SYSTEM MIGHT NEED MODIFICATIONS
C
        DEFINE FILE IIN( N, 256, U, INEXT )      ! INEXT WILL POSITION TO THE
                                     ! NEXT AVALIABLE RECORD
        WRITE(6,3)
3       FORMAT('CALCULATION CONSTANT?')
C       SEE APPENDIX B FOR FURTHER DETAIL
        READ(6,4) C                          ! SIGMA = C * DF/DF
4       FORMAT(E12.5)
        INEXT = 1                            ! MAKE SURE IT STARTS READING FROM
C                                            BLOCK 1
        RETURN
        END
C
C
C       THIS ROUTINE READS IN THE NEXT WAVEFORM
C       INEXT ALWAYS POINTS TO THE NEXT RECORD AVAILABLE
C       SET ILAST <- INEXT TO REMEMBER WHERE THE LAST WAVEFORM WAS
C       SO THAT AFTER THE CALCULATIONS THE INFORMATION CAN BE
C       WRITTEN BACK TO THE HEADER
C       READ IN THE 1'ST BLOCK FIRST BECAUSE EACH WAVEFORM IS AT
C       LEAST 1 BLOCK LONG
C       FIND OUT HOW LONG THIS WAVEFORM IS AND READ IN THE REST
C
        SUBROUTINE READIN
C       COMMON DEFINITIONS
        COMMON /DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
C       EQUIVALENCE THE 1'ST 256 WORDS
        INTEGER IBUF(256),JBUF(256)
        EQUIVALENCE (IBUF(1),IDATA(1)),(NBLK,IDATA(127))
C       SET ILAST = INEXT
        ILAST = INEXT
C       READ IN THE 1'ST BLOCK
        READ(IIN'INEXT) IBUF                 ! READ IN 256 WORD RECORD
        IF( NBLK .EQ. 1 ) RETURN             ! ONLY 1 BLOCK LONG
C       OTHERWISE, READ IN THE REST OF THE WAVEFORM
        ISTART = 256                         ! THERE ARE 256 POINTS ALREADY
                                             ! START AT THE 257 TH POINT
        DO 100 I = 1,NBLK - 1                ! DO THE REST
        READ(IIN'INEXT) JBUF                 ! READ IN 256 WORDS A TIME
C       COPY JBUF INTO IDATA AT THE RIGHT BOUNDARY
        DO 101 J = 1,256
101     IDATA(J + ISTART) = JBUF(J)
        ISTART = ISTART + 256                ! SET FOR THE NEXT 256 WORD BOUNDARY
100     CONTINUE
C       THE WHOLE WAVEFORM IS IN IDATA(*), CAN RETURN NOW
        RETURN
        END
C
C
C       THIS ROUTINE EXPANDS THE WAVEFORM TO A NEW SET OF LIMITS
C       AND DISPLAYS WITHOUT THE ANALOG DATA
C
        SUBROUTINE EXPAND
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
        DATA ICHAR,JCHAR /'  ','Y '/     ! UNIX PUT 'Y' IN THE LOWER BYTE
```

```
10      WRITE(6,1)
1       FORMAT('EXPAND ?')
        READ(5,2) ICHAR                    ! A <CR> MEANS NO
2       FORMAT(A1)
        IF(ICHAR .NE. JCHAR) RETURN        ! DO NOT EXPAND
C       OTHERWISE !
        WRITE(6,3)                         ! GET THE LIMITS
3       FORMAT('LOW-X, HIGH-X, LOW-Y, HIGH-Y ?')
        READ(5,4) ILX,IUX,ILY,IUY
4       FORMAT(4I10)
C       CALCULATE NEW A AND B
C       FOR DEVICE COORDINATES EXPANSION
        A = 4095./(FLOAT(IUY-ILY))
        B = -A * FLOAT(ILY)
C       DISPLAY THE EXPANDED WAVEFORM WITHOUT THE ANALOG CHANNEL
        CALL DISPR(ILX,IUX,ILY,IUY,0)
        GOTO 10                            ! AGAIN?
        RETURN
        END
C
C       THIS ROUTINE CALCULATE THE CONDUCTIVITIES AND
C       OUTPUTS THEM TO
C       THE OUTPUT FILE
C       ALSO PUT THESE VALUES BACK INTO THE HEADER AND RESETS ALL
C       NECESSARY INFORMATION
        SUBROUTINE ENDW
        COMMON /DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
C       THIS IS EQUIVALENCED TO THE HEADER
        INTEGER IBUF(256)
        EQUIVALENCE (IBUF(1),IDATA(1))
        EQUIVALENCE(ITIME,IDATA(121)),(ALTI,IDATA(41))
     $  ,(SIGP1,IDATA(49)),(SIGN1,IDATA(57)),
     $  (SIGP2,IDATA(65)),(SIGN2,IDATA(73))
C       CONVERT ITIME TO FLOATING POINT
        TIME = FLOAT(ITIME)
C       PASS IT TO THE ALTITUDE INTERPOLATING ROUTINE
        ALTI = ALT(TIME)
        I = INEXT
C       GO BACK TO THE LAST WAVEFORM AND WRITE OUT THE HEADER
        WRITE(IIN'ILAST) IBUF              ! WRITE THE HEADER BACK
C       GO FORWARD TO WHERE THE PRESENT WAVEFORM IS
        INEXT = I
        FIND(IIN'I)                        ! POSITION BACK TO THE NEXT RECORD
        A = 2.0475000E1                    ! RESET A AND B TO 0 - 200
        B = 0.0
C       WRITE OUT THE INFORMATION TO THE SYSTEM'S CONSOLE
C       AND THE FILE "PASS3.TMP"
        WRITE(IOUT,10) ITIME,ALTI,SIGP1,SIGN1,SIGP2
     $  ,SIGN2,IDATA(5),IDATA(6)
        WRITE(6,10) ITIME,ALTI,SIGP1,SIGN1,SIGP2,SIGN2,IDATA(5),
     $  IDATA(6)
10      FORMAT(I4,1X,F6.2,4(1X,E12.5),11X,A2,I3)
        RETURN
        END
```

```
; THE FOLLOWING TWO ROUTINES PROVIDE LED DISPLAY AND
; TEKTRONIX 603 DISPLAY
;
; LED AND LEDD ARE TWO ENTRY POINTS FOR DISPLAYING NUMERIC
; VALUES ON THE LED DISPLAY
; LED IS CALL FROM FORTRAN AND LEDD EXPECTS THE VALUES IN R4 ON CALLING
;
;
;
;
;
.NLIST
.MACRO  PUSH
MOV     %0,-(%6)
MOV     %1,-(%6)
MOV     %2,-(%6)
MOV     %3,-(%6)
MOV     %4,-(%6)
MOV     %5,-(%6)
.ENDM
.MACRO POP
MOV     (%6)+,%5
MOV     (%6)+,%4
MOV     (%6)+,%3
MOV     (%6)+,%2
MOV     (%6)+,%1
MOV     (%6)+,%0
.ENDM
;
.MACRO RETURN
RTS %7
.ENDM
;
.MACRO CALL A
JSR     %7,A
.ENDM
;
;
.LIST
; START
.MCALL ..V2..,.REGDEF
..V2..
.REGDEF
; DECLARE ENTRY POINTS
.GLOBL LED LEDD
.TITLE LED DISPLAY
; LPS ADDRESSES
ADBF=170402
.CSECT LEDIS
LED:    PUSH
        MOV     @2(R5),R4
        BR      START
        ; THIS IS THE WAY FORTRAN PASSES PARAMETERS
LEDD:   PUSH            ; IN R4 ALREADY
START:
        CLR     LED1
        ; CLEAR ALL 6 LED SEGMENTS
        MOV     #17,@#ADBF
        MOV     #417,@#ADBF
        MOV     #1017,@#ADBF
        MOV     #1417,@#ADBF
        MOV     #2017,@#ADBF
        MOV     #2417,@#ADBF
        ; COUNTER FOR 6 SEGMENTS
```

```
1$:     MOV     #-1,LED2
2$:     INC     LED2
        ; GET THE MODULO TO CONVERT FROM BINARY TO DECIMAL
        SUB     #10.,R4
        BGE     2$
        ADD     #10.,R4
        MOVB    R4,LED3
        MOVB    LED1,LED3+1
        ; LIGHT UP 1 SEGMENT
        MOV     LED3,@#ADBF
        INC     LED1
        MOV     LED2,R4
        TST     R4
        BNE     1$
        POP
        RETURN
LED1:   .WORD 0
LED2:   .WORD 0
LED3:   .WORD 0
;       .END LED
        ;
        ;
        .CSECT ERASVC
        ;
        ;
        ;       ERASE AND SET UP THE STORAGE SCOPE
        ;
        .SBTTL ERASE
        ; LPSVC STATUS
        VCST = 170416
        ; SET ERASE BIT
        ERAS = 10000
        ; Y MODE, FAST INTENSIFY
        VCRDY = 2010
        .GLOBL ERASE
        ; NAME CALLED BY FORTRAN
        ; ENTRY POINT
ERASE:  PUSH
        MOV     #ERAS,@#VCST
        MOV     #VCRDY,@#VCST
1$:     TSTB    @#VCST
        BPL     1$
        POP
        RETURN
        ;
        ;
        ; ROUTINE TO INTENSIFY AN ADDRESSED POINT ON THE 603
        ; X-Y COORDINATES ARE PASSED FROM FORTRAN
        .CSECT INTE
        .SBTTL  INTENSIFY A POINT
        ; ENTRY POINT
        .GLOBL  INTEN
        ; LPSVC X COORDINATE
        ; LPSVC Y COORDINATE
        VCX = VCST+2
        VCY = VCST+4
INTEN:  PUSH
        ; GET X AND Y VALUE INTO THE LPSVC REGISTERS
        MOV     @2(R5),@#VCX
        MOV     @4(R5),@#VCY
        ; TEST FOR READY BIT
2$:     TSTB    @#VCST
        BPL     2$
        POP
        RETURN
```

```
        .END
C       THIS ROUTINE DISPLAYS THE WAVEFORM CONTAINED IN IDATA(*)
C       ACCORDING TO THE LIMITS EXPRESSED BY IX1,IX2,IY1,IY2
C       IFLAG IS USED TO INDICATE WHEATHER THE ANALOG CHANNEL
C       CHANNEL 1 OF THE LPS ADC IS TO BE DISPLAYED OR NOT
C
        SUBROUTINE DISPR(IX1,IX2,IY1,IY2,IFLAG)
        COMMON/DATA/ IDATA(4096)
        COMMON/PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
C       THIS IS TO HOLD THE VERTICAL INTERVALS
        DIMENSION IFF(9)
C       CURRENTLY IS MARKED AT EVERY 20 HZ INTERVAL
        DATA IFF/ 20,40,60,80,100,120,140,160,180/
C       THE NUMBER OF POINTS
        N = IX2 - IX1 + 1
C       SKIP THE HEADER
        IST = 129 + 2 * (IX1 - 1)
C       ERASE THE SCOPE
        CALL ERASE
        DO 100 I = 1,9
C       DRAW THE VERTICAL GRIDS
        IY = IFF(I)
        IF( IY .LE. IY1 ) GOTO 100
        IF( IY .GE. IY2 ) GOTO 101
        IY = INT( A * FLOAT(IY) + B )
        DO 102 J = 1,4096,25
        IX = J - 1
102     CALL INTEN(IX,IY)
100     CONTINUE
C       SUM UP THE TOTAL TIME FOR HORIZONTAL EXPANSION
101     TTOTAL = 0.0
        I = IST
        DO 106 J = 1,N
        TTOTAL = TTOTAL + COUNT(IDATA(I))
106     I = I + 2
        TTOTAL = 1.0E-5 * TTOTAL
C       GET THE SCALING FACTOR IN THE X AXIS
        D = 4096.0/TTOTAL
        T = 0.0
C       DRAW OUT HORIZONTAL GRIDS
C       AND DATA POINTS
C       MARKING IS DONE ON EVERY 50 POINTS
107     DO 104 I = 1,50
        CNT = COUNT(IDATA(IST))
        F = 1.0E5/CNT
        T = T + 1.0E-5*CNT
        Y = A * F + B
        IF( Y .LE. 0.0 .OR. Y .GE. 4095.0 ) GOTO 105
        IX = INT(( D * T ))
        IY = INT(Y)
        CALL INTEN(IX,IY)
C       SKIP THE ANALOG POINT
105     IST = IST + 2
        N = N - 1
        IF( N .EQ. 0 ) GOTO 110
104     CONTINUE
C       THIS PART REALLY DRAWS OUT THE HORIZONTAL LINE
        DO 108 I = 1,4096,25
        IY = I - 1
108     CALL INTEN(IX,IY)
        GOTO 107
C       EXPAND LATER TO DRAW OUT THE ANALOG CHANNEL ALSO
110     RETURN  ! EXPAND LATER
        END
```

```
C
C
C
C       THIS FUNCTION CONVERT THE COUNT INTO A REAL NUMBER
C       SINCE THE COUNT USES FULL 16 BITS, IF THE COUNT
C       EXCEED 15 BITS, IT WILL BE VIEWED AS A NEGATIVE INTEGER
C       BY FORTRAN, THEREFORE THE PROGRAM HAS TO CHECK FOR THAT
C       NEGATE THE NUMBER AND ADD 32767 TO IT
        REAL FUNCTION COUNT(ICNT)
        IF( ICNT .LE. 0 ) GOTO 1
        COUNT = FLOAT(ICNT)
        RETURN
1       COUNT = FLOAT(-ICNT) + 32767.0
        RETURN
        END
C
C
C       THIS ROUTINE EXPONENTIALLY INTERPOLATES THE ALTITUDE
C       FROM THE TIME GIVEN
C       ALTITUDE DATA IS KEPT IN COMMON FROM WHICH IS READ IN
C       FROM THE RADAR DATA FILE
C       THE NUMBER OF RADAR DATA POINTS IS THERE ALSO
        REAL FUNCTION ALT(TIME)
        COMMON /ALTITU/ NALT,T(100),Z(100)
        IF( TIME .LT. T(1)) GOTO 10
        DO 100 I = 2,NALT
C       FIND THE RIGHT VALUES
        IF( T(I) .GT. TIME ) GOTO 11
100     CONTINUE
11      T1 = T(I-1)
        T2 = T(I)
        Z1 = Z(I-1)
        Z2 = Z(I)
C       LINEAR INTERPOLATE NOW
        ALT = Z1 + (Z2-Z1)/(T2-T1)*(TIME-T1)
        RETURN
10      WRITE(6,1)
1       FORMAT('ERROR IN ALTITUDE')
        ALT = -1.0
        RETURN
        END
C
C       THIS ROUTINE PERFORMS WEIGHTED LEAST SQUARE STRAIGHT LINE FIT
C       FOR THE SLOPE OF THE WAVEFORM
C       BREAK POINTS ARE INPUTTED FROM THE TERMINAL
C       FIRST TRAIL OF THE FIT ASSIGNS EQUAL WEIGHTS(1.0) TO ALL THE POINTS
C       WITHIN THE INTERVAL
C       SECOND TRIAL ASSIGNS WEIGHTS BETWEEN 0.0 AND 1.0 TO THE POINTS
C       ACCORDING TO THEIR RESIDUE BETWEEN THE FIRST FIT AND THE ACTUAL
C       VALUE
C       THIRD TRIAL WILL DO THE SAME THING TO FINER TUNE THE LINE
C       THE FITTED LINE IS THEN DISPLAYED ON THE SCOPE
C       CONDUCTIVITY VALUES ARE THEN CALCULATED AND INSERTED INTO THE HEADER
C       THE PROCESS IS REPEATED FOR NEGATIVE CONDUCTIVITY AND MORE
C       IF THERE WERE MORE THAN ONE SET OF CONDUCTIVITY VALUES
C
C       DATA BASE DEFINATION
C
C
C
C
        SUBROUTINE SLOPE
        COMMON/DATA/ IDATA(4096)
        COMMON /PARM/ IIN,IOUT,INEXT,ILAST,ISIZ,A,B,C,D
```

```
        REAL    SIG(8,4)                    ! THIS IS THE SIGMA BLOCK FROM HEADER
        LOGICAL FLAG                        ! FLAG TO INDICATE POS OR NEG
C       EQUIVALENCE IT TO THE HEADER
        EQUIVALENCE (SIG(1,1),IDATA(49))
        DATA ICHAR,JCHAR/ '  ','Y '/        ! UNIX PUT IN THE LOWER BYTE
        FLAG = .FALSE.                      ! FALSE = POSITIVE; TRUE = NEGATIVE
C       SEE IF EXPANSION IS NEEDED
150     CALL EXPAND
C       GET THE BREAK POINTS
        WRITE(6,3)
3       FORMAT('LOW-X, HIGH-X ?')
        READ(5,4) IX1,IX2
4       FORMAT(2I10)
        IF( IX1 .EQ. 0 ) GOTO 111         ! A <CR> = 0, SEE IF NOW IS LOOKING
C       FOR NEGATIVE COND. IF YES, SIGMA+ = SIGMA-
C       IS THE LIMITS PUT IN CORRECT
        IF( IX2 .GT. IX1 ) GOTO 1
        WRITE(6,2)
2       FORMAT( ' ERROR IN LIMIT ')
        GOTO 150
C       NUMBER OF POINTS
1       N = IX2 - IX1 + 1           ! TAKE POINTS FROM 1X1 - IX2 INCLUSIVE
        JSTART = 129 + 2*(IX1 - 1) ! SKIP THE HEADER AND GO TO THE RIGHT START
        INDEX = 1                   ! WHICH PASS
        II = 1                      ! FOR FILLING IN SIGMA VALUES
        EPS = 1.0                   ! TOLERANCE FOR RESIDUE
        TS = 0.0                    ! STARTING TIME
C
C       HAVE TO SKIP THE DATA BETWEEN POINT 1 AND IX1 AND CALCULATE THE TIME
C       ELAPSED
        DO 110 I = 129,JSTART,2 ! POINTS ARE IN PAIRS
C       SUM UP THE COUNTS
110     TS = TS + COUNT(IDATA(I))
        TS = 1.0E-5 * TS            ! CONVERT TO TIME ELIPSED
5       T = TS                      ! T STARTS FROM TS
C       STARTING INDEX
        J = JSTART
        FREQI = 0.0                 ! SUM FI
        WI = 0.0                    ! SUM WI
        TF = 0.0                    ! SUM TI * FI
        TI = 0.0                    ! SUM TI
        TI2 = 0.0                   ! SUM TI ** 2
        DO 100 I = 1,N  ! N POINTS TOTAL , START FROM THE J TH
C       SUM UP THE COUNTS
        CNT = COUNT(IDATA(J))
        F = 1.0E5/CNT               ! CONV TO FREQUENCY
        T = T + 1.0E-5 * CNT        ! TIME
        GOTO (101,102,102,103), INDEX   ! WHICH PASS IT IS
101     W = 1.0                     ! FIRST TIME WEIGHT = 1 FOR ALL POINTS
        GOTO 105
102     R = ABS(P*T + Q - F)        ! RESIDUE
        IF( R .LE. EPS ) GOTO 107
C       CALCULATE NEW WEIGHT
        W = EPS/R
        GOTO 105                    ! CAL WEIGHT
107     W = 1.0
C       SUM UP ALL INFORMATION
C       NOTE THAT THEY HAVE ALL BEEN MODIFIED BY WEIGHTS
C       THE 1'ST PASS ALL WEIGHTS ARE ASSIGNED 1.0
C       FOR SUCCESSIVE PASSES THE WEIGHTS ARE MODIFIED
C       ACCORDING TO THEIR RESIDUES
105     WI = WI + W                 ! SUM WEIGHT
        TI = TI + W * T             ! SUM TI
        TI2 = TI2 + W * T ** 2      ! SUM TI ** 2
```

```
          TF = TF + W * T * F      ! SUM T * F
          FREQI = FREQI + W * F    ! SUM FREQUENCY
          J = J + 2                ! DATA ARE STORED IN PAIRS
100       CONTINUE
          DET = WI * TI2 - TI ** 2          ! DETERMINT
C         FIND SLOPE AND INTERCEPT
          P = (WI * TF - TI * FREQI)/DET
          Q = (TI2 * FREQI - TI * TF)/DET
          INDEX = INDEX + 1                 ! NEXT PASS
          GOTO 5
C         DISPLAY THE FINAL FITTED LINE
103       IF( FLAG) GOTO 200
C         RESTRICT TO 200 POINTS FOR LINE GENERATION
          DT = 20.48 / D
          T = 0.0
          DO 120 I = 1,200
          F = P*T + Q
          Y = A*F + B
          IF( Y .LE. 0.0 ) GOTO 120
          IF( Y .GE. 4095.0) GOTO 121
          IY = INT(Y)
          IX = INT(D*T)
          CALL INTEN(IX,IY)
120       T = T + DT
121       GOTO 201
C         CHOOSE BETWEEN WHICH DIRECTION TO GENERATE THE LINE FOR
C         BEST EFFECT
200       TTOTAL = 4096.0/D
          DO 130 I = 1,201
          F = FLOAT(I-1)
          T = (F-Q)/P
          IF( T .LE. 0.0) GOTO 130
          IF( T .GE. TTOTAL) GOTO 201
          Y = A * F + B
          IX = INT( T * D )
          IY = INT(Y)
          CALL INTEN(IX,IY)
130       CONTINUE
201       WRITE(6,9)
9         FORMAT('TRY AGAIN?')
          READ(5,7) ICHAR
          IF( ICHAR .EQ. JCHAR ) GOTO 150
C         CALCULATE THE RIGHT SIGMA VALUES AND PUT THEM IN THE RIGHT PLACE
          IF( FLAG ) GOTO 112               ! NEGATIVE
          SIG(1,II) = P * C                 ! SIGMA PLUS
          FLAG = .NOT. FLAG                 ! UPDATE FLAG
          GOTO 150                          ! TRY FOR SIGMA MINUS
112       SIG(5,II) = P * C                 ! SIGMA MINUS
          FLAG = .NOT. FLAG                 ! UPDATE FLAG
          II = II + 1                       ! IN CASE THERE IS MORE THAN 1 SIGMA VALUE
C         AT MOST 4 SET OF SIGMA VALUES ARE ALLOWED
          IF( II .GT. 4) GOTO 141
          WRITE(6,6)
6         FORMAT('ANOTHER SET?')
          READ(5,7) ICHAR
7         FORMAT(A1)
          IF( ICHAR .EQ. JCHAR ) GOTO 150
C         OTHERWISE, FILL IN THE REST WITH OS
          DO 140 I = II,4
          SIG(1,I) = 0.0
          SIG(5,I) = 0.0
140       CONTINUE
141       CALL ENDW
          RETURN
```

```
111     IF( FLAG ) GOTO 112
        RETURN
        END
C       PROGRAM TO EXTRACT DATA FROM HEADER
C       INTO BASIC-PLUS VIRTUAL ARRAY (DISK FILE) FORMAT
C       FOR PLOTTING ON THE PDP 11/45
C
C       VIRTUAL ARRAY IS ORGANIZED AS FOLLOW:
C
C       IN BASIC --
C               OPEN "XXXXX" AS FILE #?
C               DIM #?, X(300),Y(300)
C               REM X(0) CONTAINS THE NUMBER OF COORDINATES
C               REM Y(0) IS NOT USED
C               REM SINCE BASIC STARTS ARRAY AT OTH ELEMENT
C               REM SO THERE ARE ACTUALLY 301 PAIRS OF
C               REM POINTS IN THE FILE
C               REM THIS NUMBER MUST BE MAINTAINED FOR PROPER
C               REM DATA ALIGNMENT
C
C
C       IN FORTRAN --
C               DISK FILE CAN BE ADDRESSED BY BINARY I/O
C               AND DEFINE FILE STATEMENT
C
C               CALL ASSIGN(??????????????) TO ASSIGN
C               LOGICAL UNIT NUMBER FOR FILE(S)
C
C               DEFINE FILE ?(602,4,U,II)
C               602 RECORDS (301 XS + 301 YS)
C               EACH 4 WORDS LONG (BASIC PUTS EVERYTHING IN
C               DOUBLE PRECISION)
C
C               READ (?'IREC) FOR READ
C               WRITE(?'IREC) FOR WRITE
C               FIND(?'IREC) FOR POSITIONING
C
C               SEE PDP 11/RT-11 FORTRAN MANUAL FOR REF.
C
C
C
C       COMMON DECLARATION
        COMMON /DATA/ IDATA(4096)
        COMMON/ PARM/ IIN,IOUT, INEXT,ILAST,ISIZ,A,B,C,D
        EQUIVALENCE (NPTS,IDATA(125)),(ALT,IDATA(41)),(SIGPOS,
      $ IDATA(49)),(SIGNEG,IDATA(57))
C       THEY ARE USED TO HOLD ALTITUDE, SIG+, SIG-
        REAL*8  Z,X,Y
C
C       LOGICAL UNIT NUMBER 6 IS INPUT TERMINAL
C       RT-11 DEFAULT TERMINAL TO 7
        CALL ASSIGN(6,'TT:/N')
C
        WRITE(6,1)
1       FORMAT('EXTRACT DATA FROM HEADER')
        WRITE(6,2)
2       FORMAT('INPUT DATA FILE NAME ?',/)
C       THE PASS2 FILE NAME
        CALL ASSIGN(19,'??????.???',-1,'RDO')
        IIN = 19
C
        WRITE(6,3)
C       THE 3 OUTPUT FILE NAMES
C       SIGMA+, SIGMA-, SIGMA+ = SIGMA-
C
```

```
3       FORMAT('OUTPUT FILE 1 -- POS ?',/)
        CALL ASSIGN(20,'??????.???',-1,'NEW')
C
        WRITE(6,4)
4       FORMAT('OUTPUT FILE 2 -- NEG ?',/)
        CALL ASSIGN(21,'??????.???',-1,'NEW')
C
        WRITE(6,5)
5       FORMAT('OUTPUT FILE 3 -- COM ?',/)
        CALL ASSIGN(22,'??????.???',-1,'NEW')
C
        WRITE(6,6)
C       THIS IS REQUIRED BY RT-11 DEFINE FILE STM
6       FORMAT('INPUT FILE SIZE ?')
        READ(5,11) ISIZ
11      FORMAT(I4)
C
C
        DEFINE FILE 19(N,256,U,INEXT)
        DEFINE FILE 20(602,4,U,I1)
        DEFINE FILE 21(602,4,U,I2)
        DEFINE FILE 22(602,4,U,I3)
C
        INEXT = 1
C       SINCE FORTRAN STARTS ARRAY AT 1
C       THEREFORE BASIC ARRAY(1) = FORTRAN ARRAY(2)
C
        IPOS = 2
        INEG = 2
        ICOM = 2
C
C       START READING
C
100     CALL READIN
        IF(INEXT .GE. ISIZ) GOTO 1000
        IF(NPTS .LE. 10) GOTO 100
        IF( ALT .LE. 0.0) GOTO 100
        IF(SIGPOS .EQ. SIGNEG) GOTO 101
C
C       CONVERT TO DOUBLE PRECISION
        Z = DBLE(ALT)
        X = DBLE(SIGPOS)
        Y = DBLE(SIGNEG)
C
C       WRITE THEM OUT TO DISK
        WRITE(20'IPOS) X
        WRITE(20'IPOS+301) Z
        WRITE(21'INEG) Y
        WRITE(21'INEG+301) Z
C
C       INCREMENT THE COUNTER
        IPOS = IPOS + 1
        INEG = INEG + 1
        GOTO 100
C
C
C       IF THEY WERE EQUAL
101     Z = DBLE(ALT)
        X = DBLE(SIGPOS)
C
        WRITE(22'ICOM) X
        WRITE(22'ICOM+301) Z
        ICOM = ICOM + 1
C
```

```
      GOTO 100
C
C
C     WRITE OUT THE NUMBER OF POINTS
C
1000  WRITE(20'1) DBLE(FLOAT((IPOS-2)))
      WRITE(21'1) DBLE(FLOAT((INEG-2)))
      WRITE(22'1) DBLE(FLOAT((ICOM-2)))
      END
```

END
DATE
FILMED
10-8
DTIC